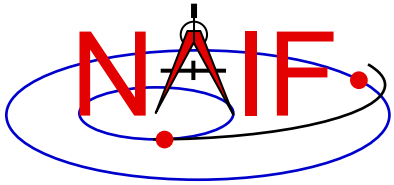


**Navigation and Ancillary Information Facility**

# **Ephemeris Subsystem SPK**

**Focused on reading SPK files**

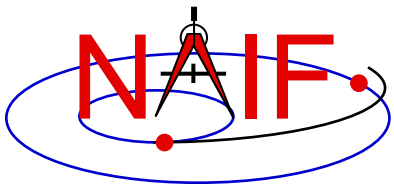
**October 2022**



# First... clear your mind!

Navigation and Ancillary Information Facility

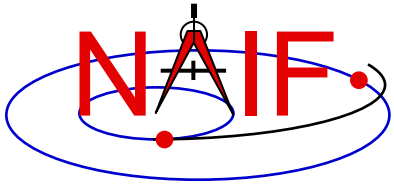
- **SPK is probably unlike any previous ephemeris or trajectory representation you've used or heard about.**
- **We think you'll find it to be more capable than other ephemeris system architectures.**
  - As such, it's also a bit more complicated to grasp.
- *Don't panic! Shortly you'll be reading SPK files like a pro.*



---

Navigation and Ancillary Information Facility

# Overview of SPICE Ephemeris Data

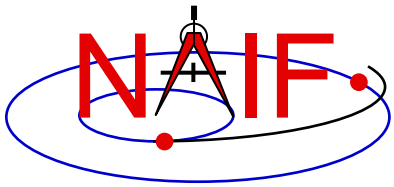


# A Picture is worth ...

---

Navigation and Ancillary Information Facility

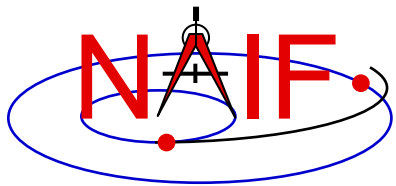
- **We'll start with a mostly pictorial view of ephemeris data and SPK files, just to ease you into this topic.**



# SPICE Ephemeris Data

Navigation and Ancillary Information Facility

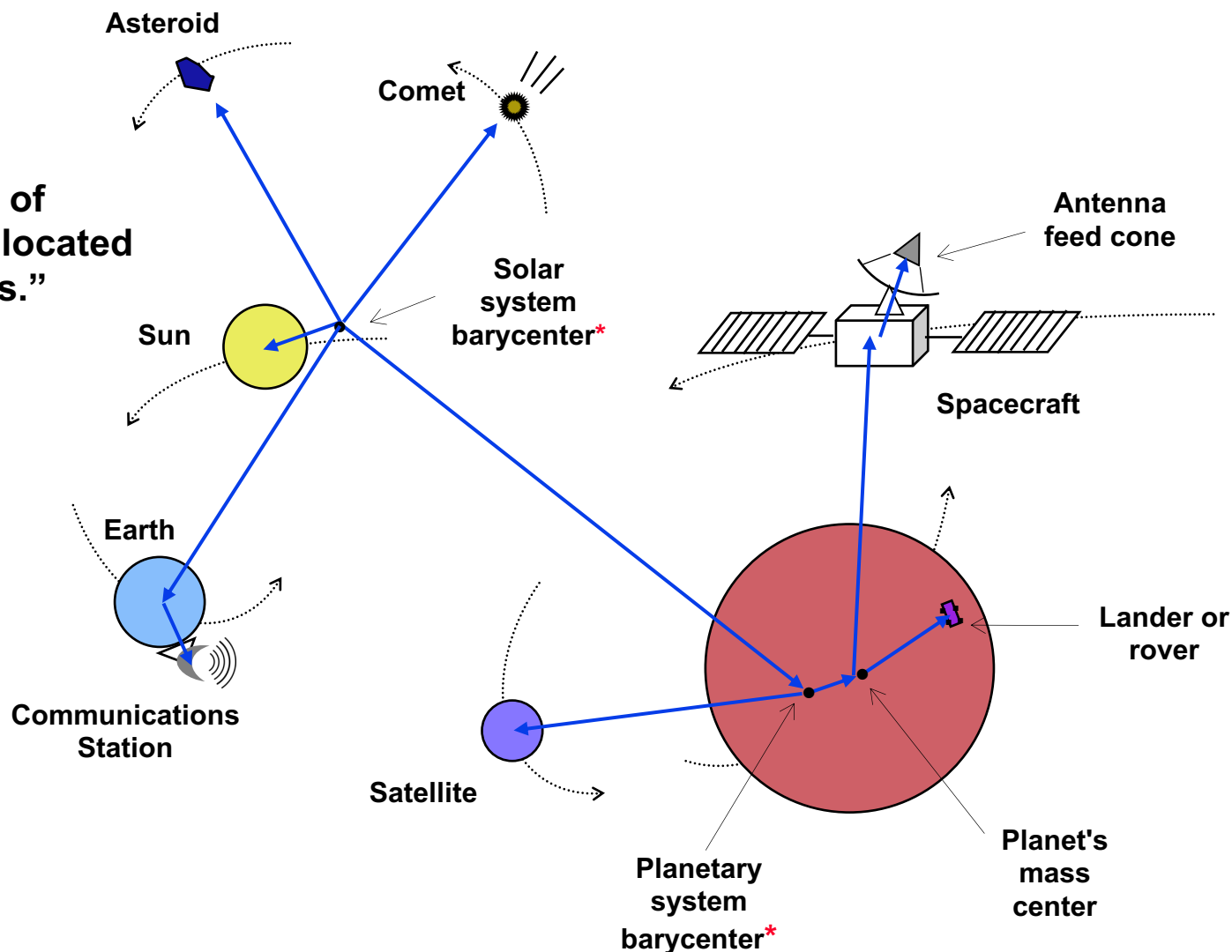
- **An SPK file contains ephemeris (trajectory) data for "ephemeris objects."**
  - “Ephemeris” means position and velocity as a function of time
    - » Position + velocity is often referred to as “state”
- **“Ephemeris objects” are spacecraft, planets, satellites, comets and asteroids.**
  - **But the following are also ephemeris objects:**
    - » the center of mass of our solar system (solar system barycenter)
    - » the center of mass of a planet/satellite system (planet barycenter)
    - » a rover on the surface of a body
    - » a camera on top of a mast on a lander
    - » a transmitter cone on a spacecraft
    - » a deep space communications antenna on the earth

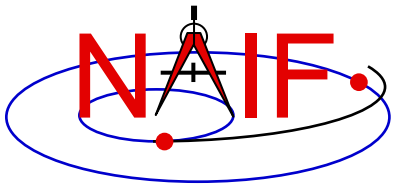


# Examples of SPICE Ephemeris Objects

Navigation and Ancillary Information Facility

The head and the tail of every **blue arrow** are located at “ephemeris objects.”





# Imagine Some Ephemeris Data

Navigation and Ancillary Information Facility

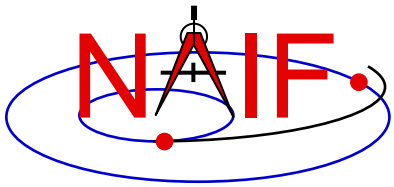
```
epoch_1, x1, y1, z1, vx1, vy1, vz1
epoch_2, x2, y2, z2, vx2, vy2, vz2
epoch_3, x3, y3, z3, vx3, vy3, vz3
epoch_4, x4, y4, z4, vx4, vy4, vz4
..... etc. ....
..... etc. ....
epoch_n, xn, yn, zn, vxn, vyn, vzn
```

Perhaps this is an ASCII table or an Excel spreadsheet containing rows of time-tagged Cartesian state vectors.

“epoch” = time

**It may not be written inside the table or spreadsheet, but perhaps an interface agreement somehow tells you:**

- what object this ephemeris is for
- what is the name of the reference frame in which the data are given
- what is the center of motion of the object
- what time system is being used for the epochs
- what are the start and stop times of the file
  - » meaning, what are “epoch\_1” and “epoch\_n”



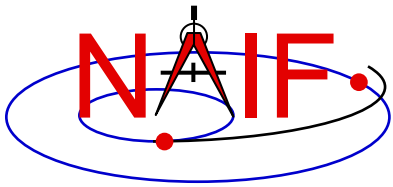
# Imagine a Simple Ephemeris File

Navigation and Ancillary Information Facility

```
epoch_1, x1, y1, z1, vx1, vy1, vz1  
epoch_2, x2, y2, z2, vx2, vy2, vz2  
epoch_3, x3, y3, z3, vx3, vy3, vz3  
epoch_4, x4, y4, z4, vx4, vy4, vz4  
..... etc. ....  
..... etc. ....  
epoch_n, xn, yn, zn, vxn, vyn, vzn
```



We'll represent that simple ephemeris data shown on the previous page as a "block" like this.



# Imagine a Simple Ephemeris File

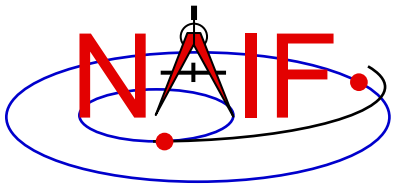
Navigation and Ancillary Information Facility

```
epoch_1, x1, y1, z1, vx1, vy1, vz1  
epoch_2, x2, y2, z2, vx2, vy2, vz2  
epoch_3, x3, y3, z3, vx3, vy3, vz3  
epoch_4, x4, y4, z4, vx4, vy4, vz4  
..... etc. ....  
..... etc. ....  
epoch_n, xn, yn, zn, vxn, vyn, vzn
```



We'll represent that simple ephemeris file as a “block” like this.

**This becomes the basis of a “segment” in an SPK file.**



# An SPK “Segment”

Navigation and Ancillary Information Facility

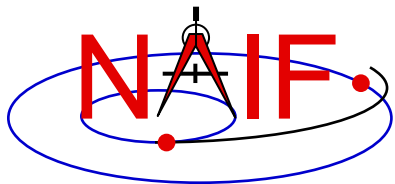
Target, Ref Frame ID, Center of Motion,  $T_{\text{start}}$ ,  $T_{\text{stop}}$

epoch\_1, x1, y1, z1, vx1, vy1, vz1  
epoch\_2, x2, y2, z2, vx2, vy2, vz2  
epoch\_3, x3, y3, z3, vx3, vy3, vz3  
epoch\_4, x4, y4, z4, vx4, vy4, vz4  
..... etc. ....  
..... etc. ....  
epoch\_n, xn, yn, zn, vxn, vyn, vzn

One segment

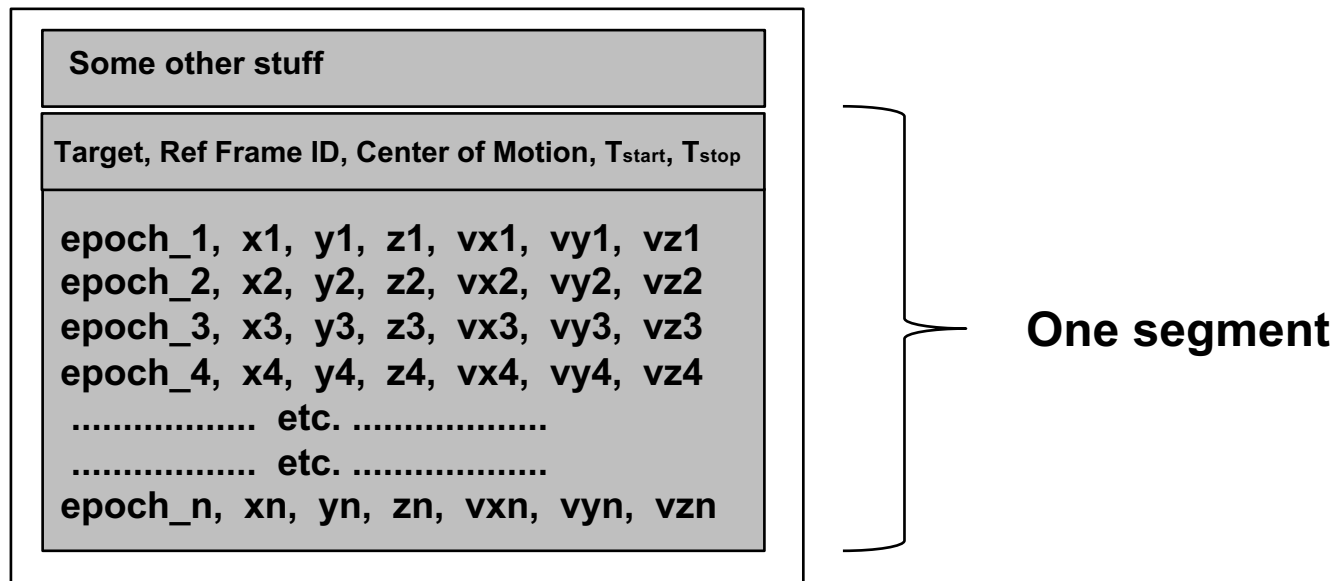
**We insert some meta-data into the segment:**

- what is the object this ephemeris is for – SPICE calls this the “target”
- what is the ID of the reference frame in which the data are given
- what is the center of motion for the target
- the start and stop times of the file,  $T_{\text{start}}$  and  $T_{\text{stop}}$ 
  - » meaning, what are “epoch\_1” and “epoch\_n”

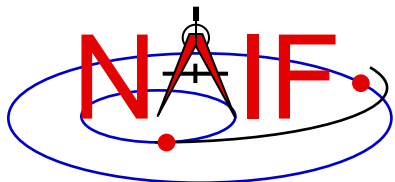


# A Simple SPK File

Navigation and Ancillary Information Facility

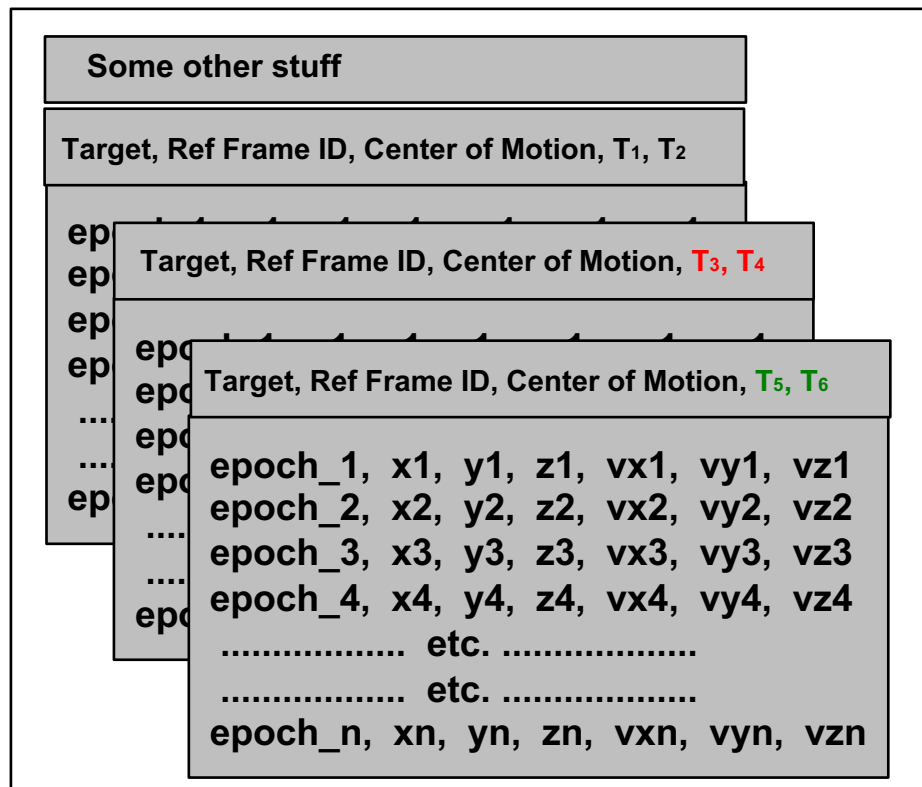


- This very simple SPK file is made up of a single segment containing ephemeris data:
  - for a single object (perhaps a spacecraft, an asteroid, or ...whatever),
  - given in a single reference frame,
  - having a single center of motion,
  - with data spanning from  $T_{\text{start}}$  to  $T_{\text{stop}}$

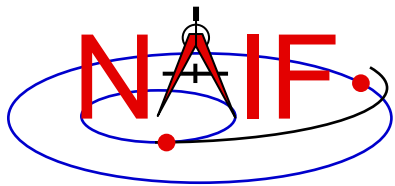


# A More Substantial SPK File

Navigation and Ancillary Information Facility

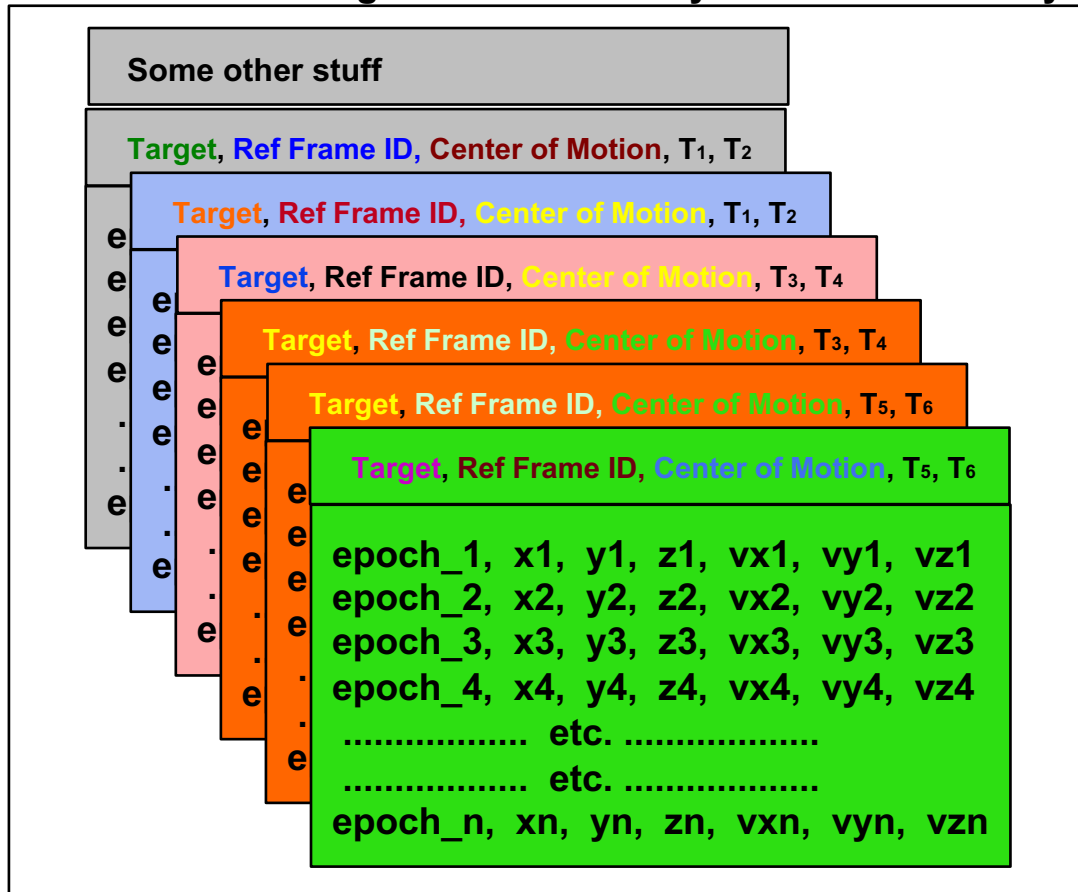


- This more substantial SPK is made up of multiple segments containing ephemeris data:
  - for a single object (perhaps a spacecraft, an asteroid, or ...???)
  - given in a single reference frame (“coordinate frame”),
  - having a single center of motion,
  - with data spanning from  $T_1$  to  $T_6$

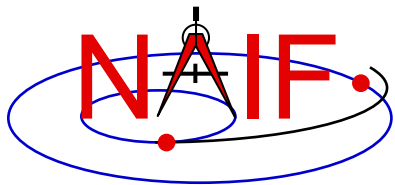


# An Even More Substantial SPK File

Navigation and Ancillary Information Facility

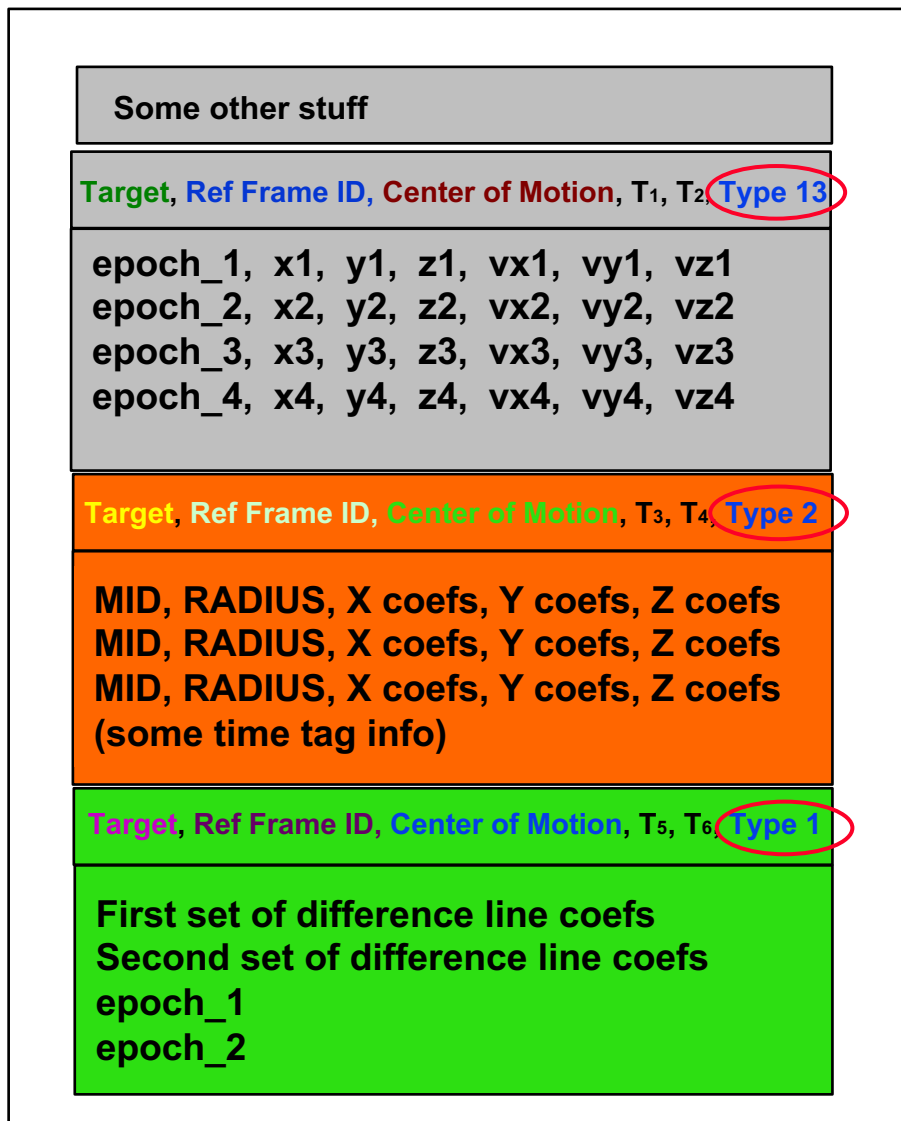


- This even more substantial SPK contains multiple segments having:
  - several objects (targets)
  - several reference frames
  - several centers of motion
  - several pairs of start and stop times

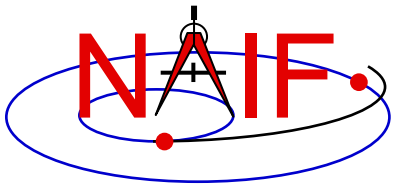


# SPK “Type” Info in each Segment

Navigation and Ancillary Information Facility



- Each segment can contain a different type of ephemeris data (as long as it's been built into the SPK subsystem). Examples:
  - Discrete state vectors
  - Chebyshev polynomials
  - Difference lines (unique to JPL)
  - Etc., etc.
- Each segment has the SPK Type stored in its meta-data record.
- Toolkit software knows how to evaluate each Type – no worries for you!



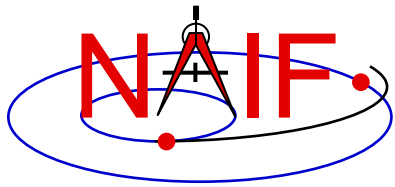
# SPK Data are Continuous Within a Segment

Navigation and Ancillary Information Facility

**Cassini**, Ref Frame ID, **Saturn bc**, T<sub>1</sub>, T<sub>2</sub>, Type 13

|          |     |     |     |      |      |     |
|----------|-----|-----|-----|------|------|-----|
| epoch_1, | x1, | y1, | z1, | vx1, | vy1, | vz1 |
| epoch_2, | x2, | y2, | z2, | vx2, | vy2, | vz2 |
| epoch_3, | x3, | y3, | z3, | vx3, | vy3, | vz3 |
| epoch_4, | x4, | y4, | z4, | vx4, | vy4, | vz4 |

- Within the time bounds (T<sub>1</sub>, T<sub>2</sub>) of a single segment, SPICE software will return a result—a state vector consisting of position and velocity—at **any** epoch... not just at the discrete epochs of the ephemeris records (epoch\_1, epoch\_2, epoch\_3, epoch\_4)
- In the example above, SPICE will return the position and velocity (the state) of the **Cassini spacecraft** relative to the **Saturn barycenter** at any time  $t$  where:  $T_1 \leq t \leq T_2$

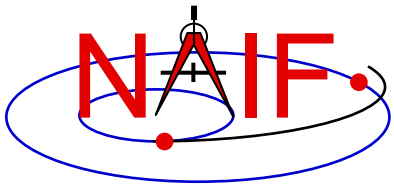


# Chaining and Frame Transformation

---

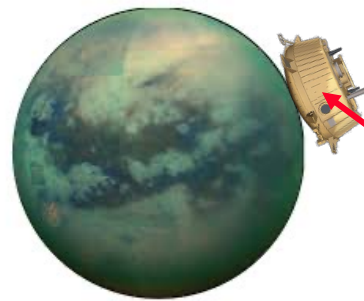
Navigation and Ancillary Information Facility

- Next we'll discuss “chaining” and “frame transformations”... features of the SPK subsystem that make it rather unique.



# Your Question

Navigation and Ancillary Information Facility



Titan

Huygens Probe

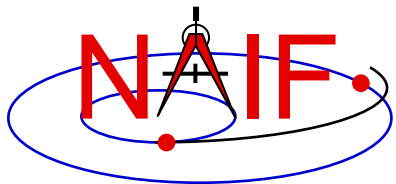


You would like to obtain the position of the Huygens Probe relative to DSS14, as shown by the **red** vector

Deep Space Station 14

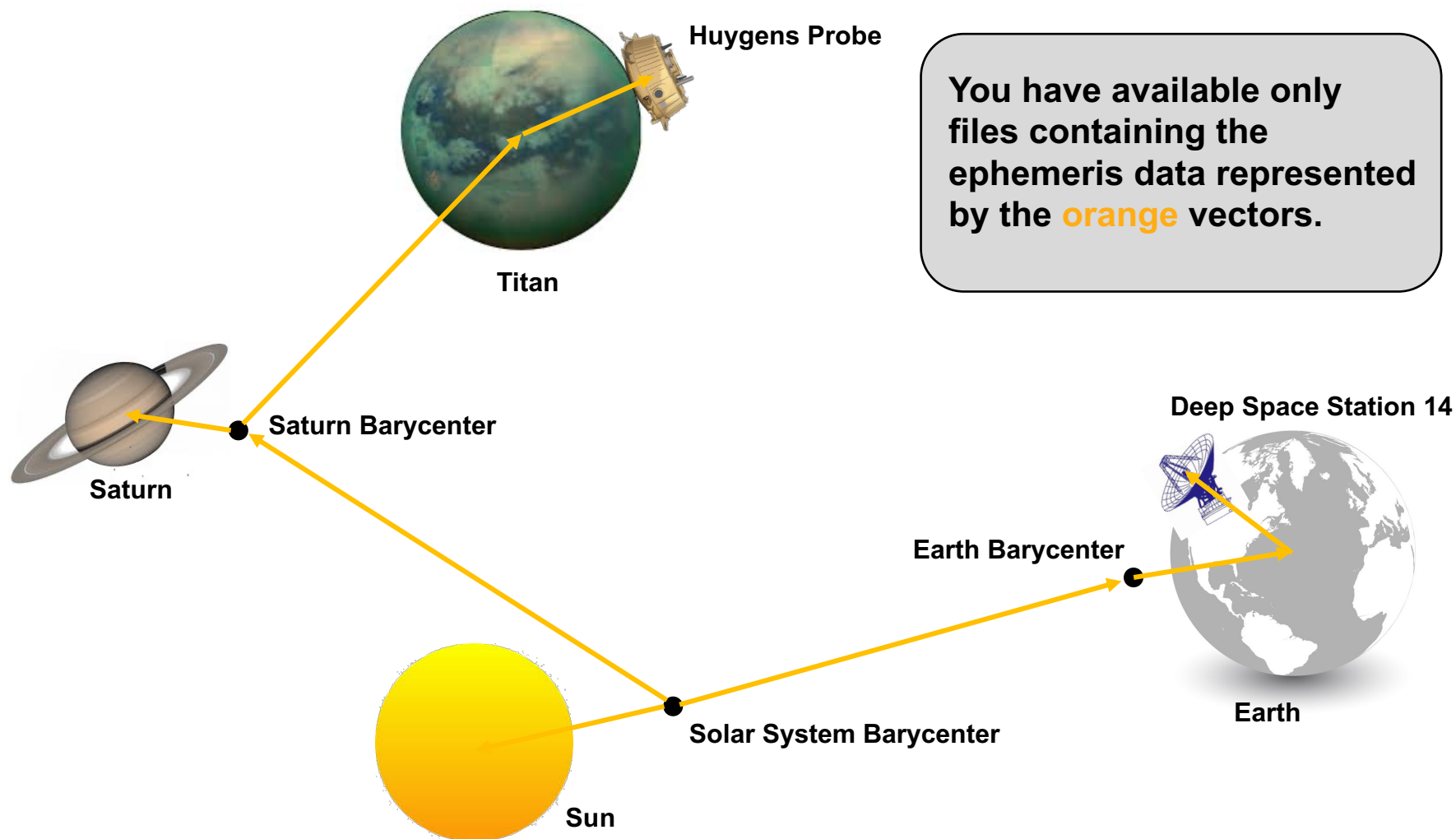


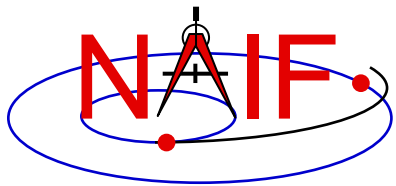
Earth



# Your Dilemma

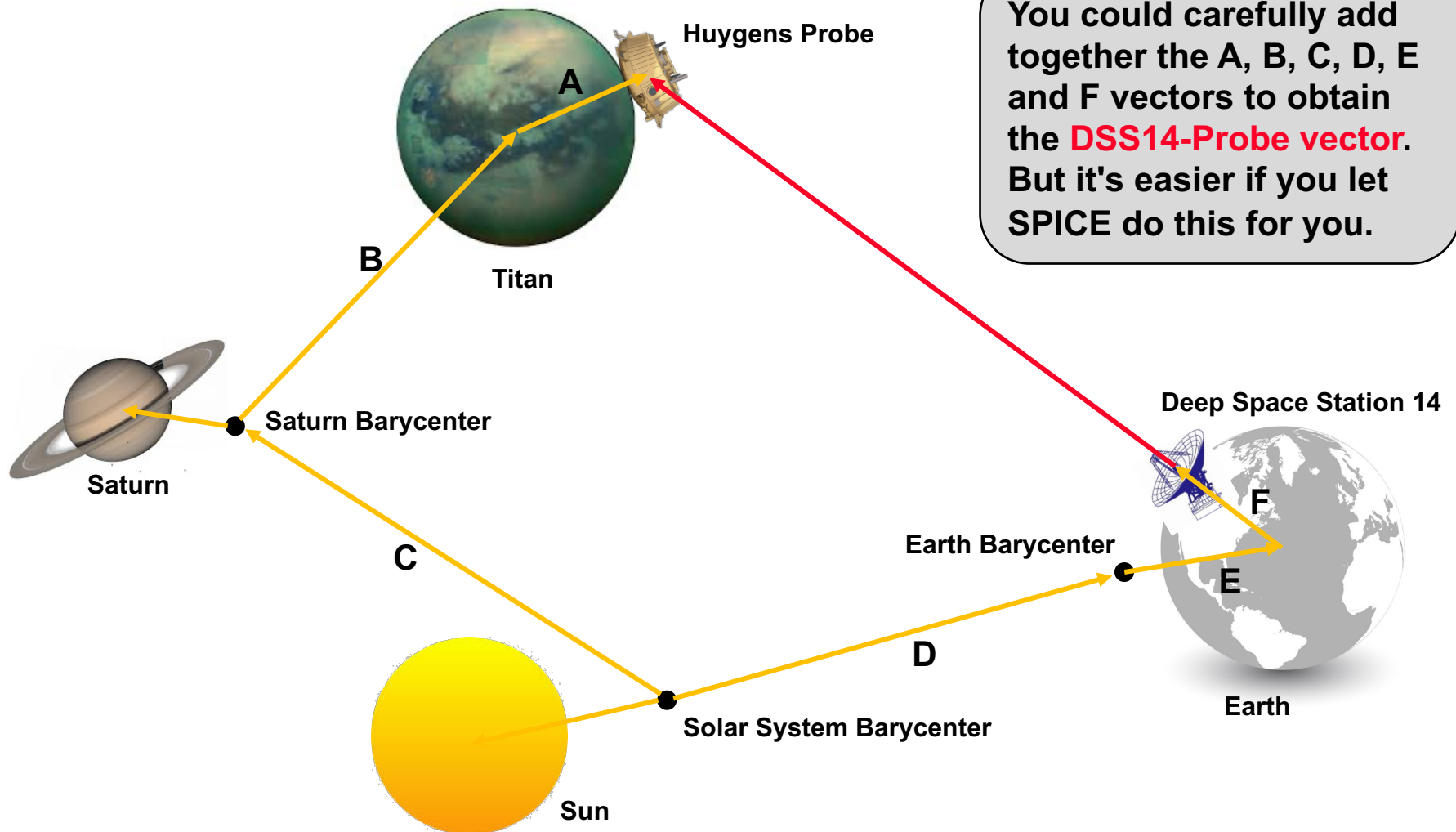
Navigation and Ancillary Information Facility



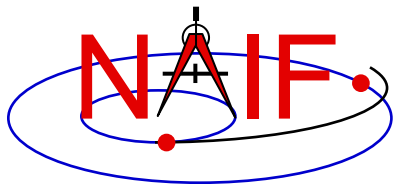


# Not a Problem

Navigation and Ancillary Information Facility



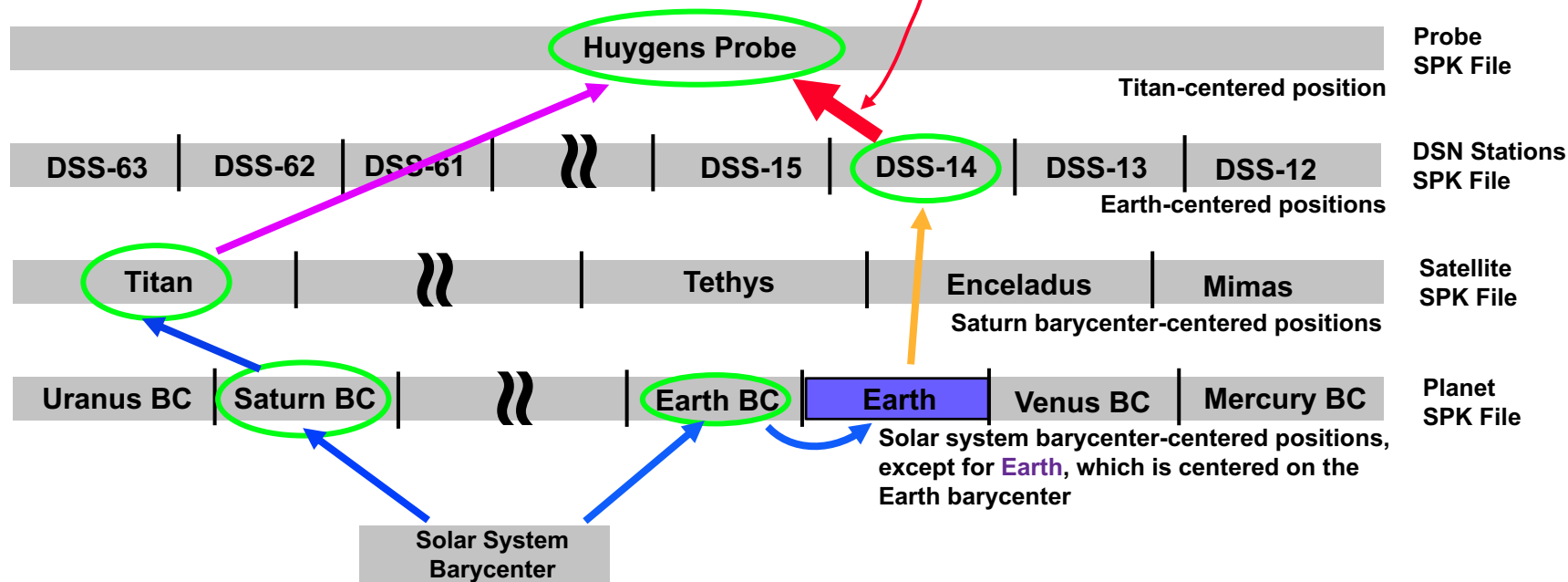
You could carefully add together the A, B, C, D, E and F vectors to obtain the **DSS14-Probe vector**. But it's easier if you let SPICE do this for you.

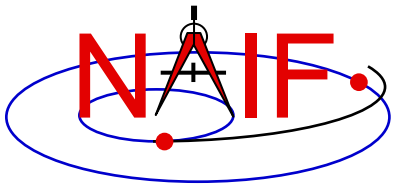


# SPICE Chains SPK Data

Navigation and Ancillary Information Facility

- **SPICE automatically searches across all loaded SPK files to find the segments needed to compute the vectors needed to obtain the result the customer has asked for. SPICE chains these together using addition and subtraction.**
  - In this example the user wants the **position** of the Huygens probe sitting on the surface of Titan as seen from Deep Space Station 14.
  - SPICE computes this by chaining the **gold**, **blue** and **violet** chunks.

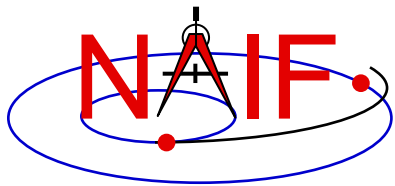




# Maybe It's Not "Simple Addition"

Navigation and Ancillary Information Facility

- **What if the A, B, C, D, E and F vectors shown two pages ago are given with respect to several different reference frames? (This is the normal situation!)**
  - Before you can add the vectors together you need to rotate them into a common reference frame.
  - This may not be very easy to accomplish on your own.
  - Once the addition is complete you may need to rotate the resultant vector into the reference frame appropriate for the job you are doing.







# SPICE Automates Frame Transformation

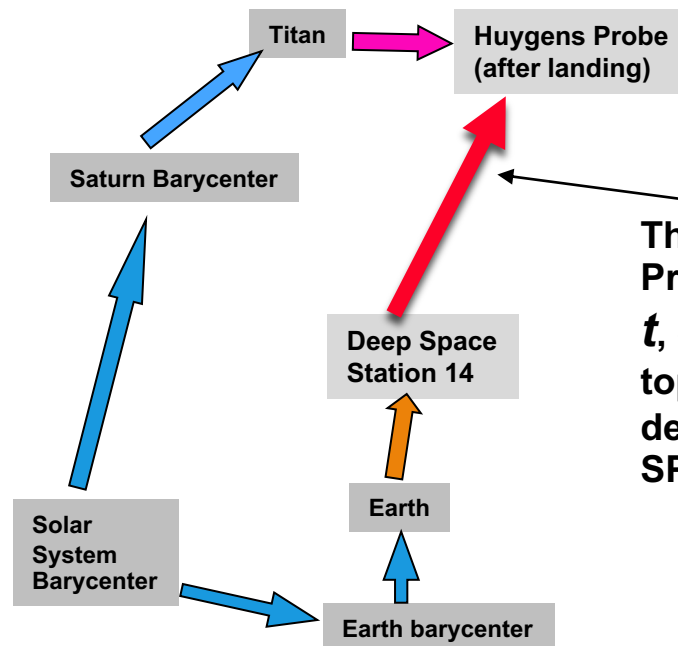
Navigation and Ancillary Information Facility

- As part of the “chaining” process just mentioned...
  - position vectors are automatically rotated into a consistent reference frame
  - the final vector is rotated into the output reference frame requested by the user

## Reference Frames Used

-  International Celestial Reference Frame (J2000)
-  Titan body-fixed frame (IAU\_TITAN)
-  International Terrestrial Reference Frame (ITRF93)
-  DSS-14 topocentric reference frame (DSS-14\_TOPO)

## Ephemeris Segments Used

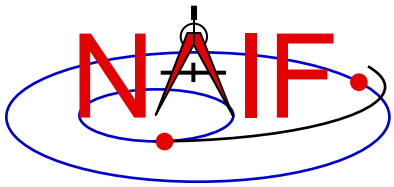


The **position** of the Huygens Probe relative to DSS-14, at time  $t$ , given in the DSS-14 topocentric reference frame, is determined using a single SPICE Toolkit API.

Fortran syntax  
used here

*A single API does it all!*

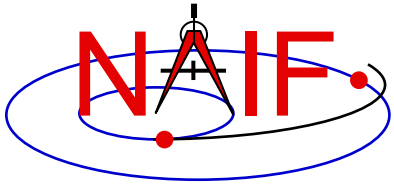
```
CALL SPKPOS ('HUYGENS_PROBE', t, 'DSS-14_TOPO', 'CN+S', 'DSS-14', POSITION, LT)
```



# SPK Segment Order and Priority

Navigation and Ancillary Information Facility

- **Within a single SPK file...**
  - The segments in an SPK file **need not be ordered** according to time or body.
  - But segment order **does imply priority**. If two segments from the same SPK file both contain data for a given target and time that satisfy a request, the SPK system selects the segment for which the **physical location** is positioned **later** in the file.
    - » The centers of motion, frames and SPK types are irrelevant to this selection.
- **If using two or more SPK files...**
  - Segments from SPK files loaded **later** have higher priority: if two segments from two different SPK files both contain data for a given target and time, the SPK system selects the segment from the SPK file that was loaded later.
    - » The centers of motion, frames and SPK types are irrelevant to this selection.



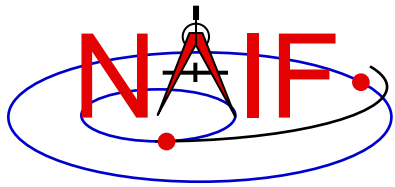
# Details

---

Navigation and Ancillary Information Facility

- **Now for some details.**
- **There's quite a lot... don't feel you need to grasp all of this immediately.**

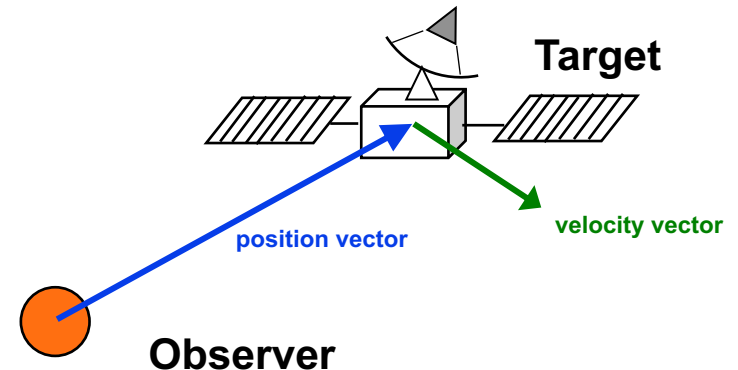
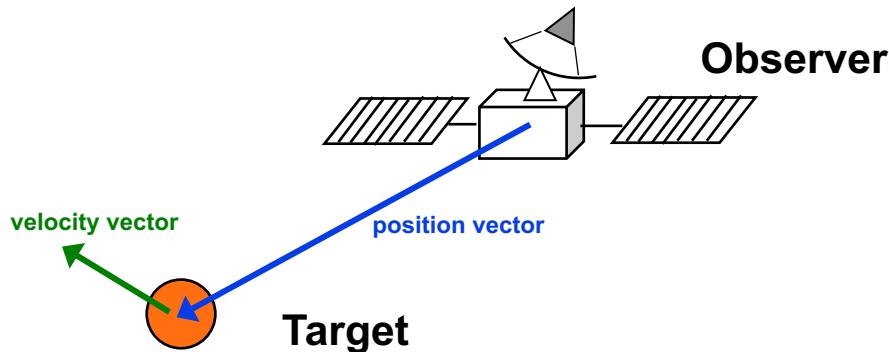




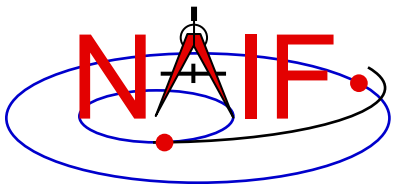
# Reading an SPK: Observers and Targets

Navigation and Ancillary Information Facility

- When you **read** an SPK file you specify which ephemeris object is to be the “target” and which is to be the “observer.”
- The SPK system returns the state of the target relative to the observer.
  - The computed **position** data point from the “observer” to the “target.”
  - The computed **velocity** is that of the “target” relative to the “observer.”



- Any ephemeris object can be a target **or** an observer!



# SPK File Coverage - 1

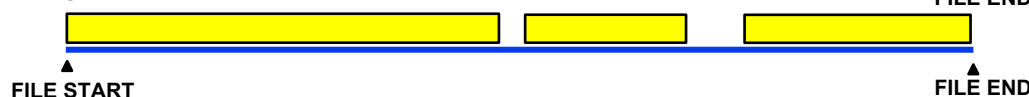
## Navigation and Ancillary Information Facility

- The time period over which an SPK file provides data for an ephemeris object is called the “coverage” or “time coverage” for that object.
  - An SPK file’s coverage for an object consists of one or more time intervals.
  - Often the coverage for all objects in an SPK file is a single, common time interval.

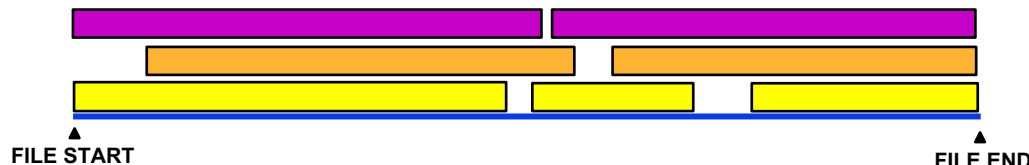
SPK file containing data for one object with no data gaps



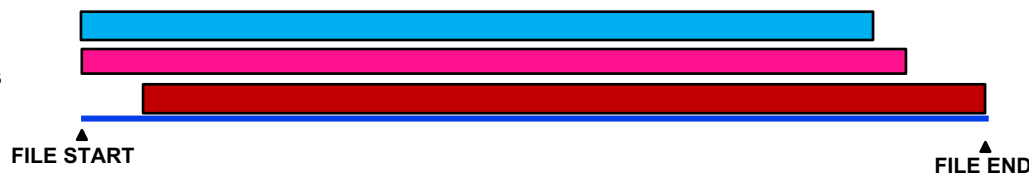
SPK file containing data for one object, with two data gaps



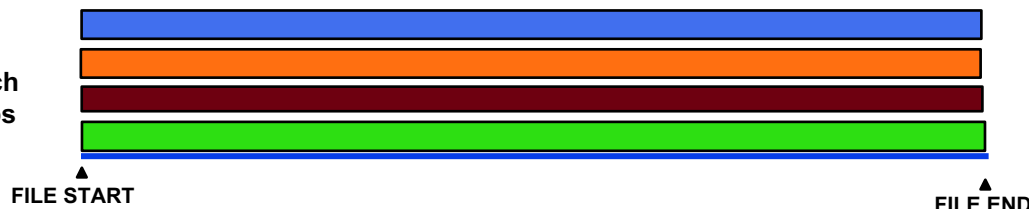
SPK file containing data for three objects, each having different data gaps

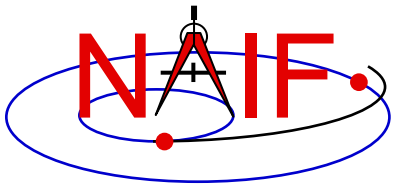


SPK file containing data for three objects, each having different coverage but with no data gaps



SPK file containing data for several objects, each having the same coverage and with no data gaps



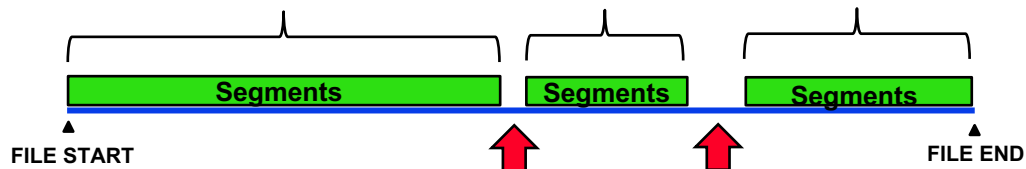


# SPK File Coverage - 2

Navigation and Ancillary Information Facility

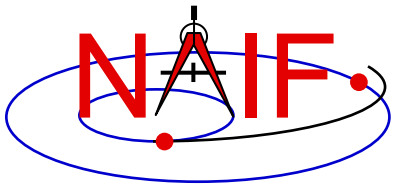
- For any request time within any time interval comprising the coverage for an object (i.e. the three green stripes shown below), the SPK subsystem can return a vector representing the state of that object relative to its center of motion.
  - The SPK system will automatically interpolate ephemeris data to produce a Cartesian state vector at the request time.
  - To a user's program, the ephemeris data appear to be **continuous** over each time interval, even if the data stored inside the SPK file are discrete.
- The SPK subsystem will *not* return a result for a request time falling within a data gap.
  - Data gaps can only occur between segments.

**“Results” will be returned by the SPK reader API for any request time falling within these three intervals.**



Note: each of the green stripes above consists of one or more segments.

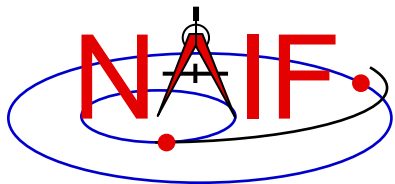
**A SPICE error message will be returned by the SPK subsystem for any request time falling within these two data gaps**



# Reference Frames Used in Writing and Reading SPKs

Navigation and Ancillary Information Facility

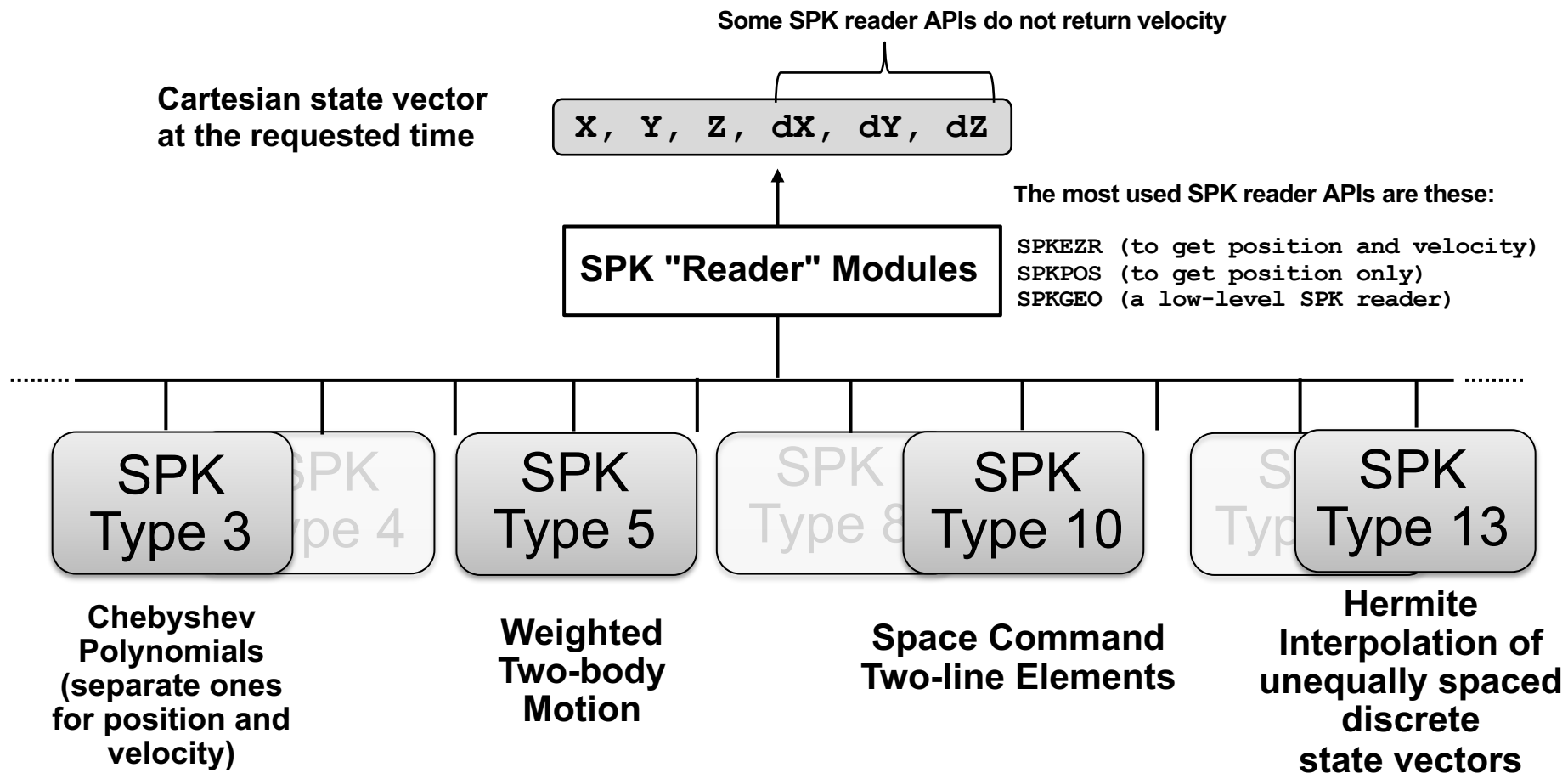
- **All ephemeris data have an associated reference frame**
  - The frame specification is input by the SPK producer and is stored in the segment's meta-data
    - » This input frame must be one “known” to the SPICE system
  - The frame may be different across segments
- **A program reading an SPK file specifies relative to what reference frame the output state or position vectors are to be given; you're not stuck with using the frame the SPK producer used**
  - This output frame you select must be known to your program
    - » “Known” means either a built-in frame (hard coded in SPICE) or one specified in a Frames Kernel
    - » The user's program may need to have access to additional SPICE data in order to construct the specified frame

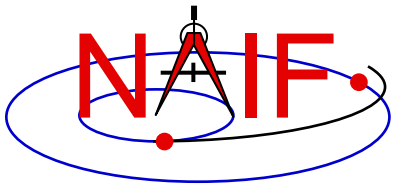


# SPK Data Type Concept

Navigation and Ancillary Information Facility

**SPK files may contain various mathematical representations of ephemeris data ("data types"), but the high-level user interfaces (SPK "readers") are type-independent:**

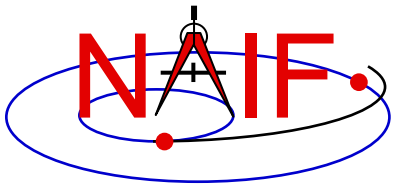




# Why Have Multiple Data Types?

Navigation and Ancillary Information Facility

- **To allow an SPK producer to choose an ephemeris representation well-suited for her/his application. For example:**
  - **Weighted two-body extrapolation (Type 5) yields compact files; may be used with sparse data. Only accurate for motion that is well approximated by the two-body model.**
  - **SPK files based on sliding-window Lagrange and Hermite interpolation (Types 9 and 13) are easy to create. Position can be made arbitrarily accurate with sufficiently small time separation of states.**
  - **Chebyshev polynomials (Types 2, 3, 14) yield the best combination of evaluation speed and accuracy. But the file creator must do more work to use these data types.**
- **To replicate data originally provided in another format.**
  - **Types 1, 2, 3, 8, 10, 14, 15, 17 and 18 were developed to enable accurate duplication of data obtained from original ephemeris developers.**

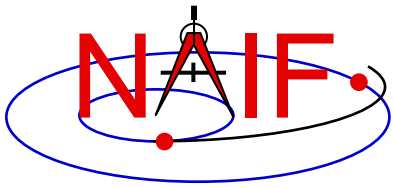


# Widely Used SPK Data Types

---

Navigation and Ancillary Information Facility

- **Type 1 (Modified divided difference arrays)**
  - Used by JPL orbit determination software for spacecraft ephemerides
- **Type 2 (Chebyshev polynomials for position, velocity given by differentiation)**
  - Used for JPL planetary ephemerides
- **Type 3 (Separate Chebyshev polynomials for position and velocity)**
  - Used for JPL satellite ephemerides
- **Type 5 (Weighted two-body extrapolation)**
  - Used for comets and asteroids, as well as for sparse data sets where a piecewise two-body approximation is acceptable
- **Type 10 (Space command two-line elements)**
  - Used for some earth orbiters
- **Types 9 and 13 (Sliding-window Lagrange and Hermite interpolation of unequally-spaced states)**
  - Used by many non-JPL ephemeris producers and by MKSPK users
- **Type 18, 19 (Sliding window Hermite or Lagrange interpolation)**
  - Used in SPKs made by ESA planetary missions (through Rosetta)



# Barycenters

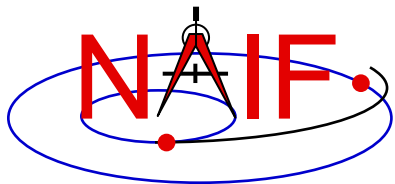
Navigation and Ancillary Information Facility

- **For planets**

- A planet and its satellites orbit the planet system's barycenter
  - » For example, the planet Jupiter (599) and each of Jupiter's satellites (501 - 5xx) orbit the Jupiter system barycenter (5)
- Because Mercury and Venus have no satellites, their barycenters (1 and 2) are at exactly the same locations as their mass centers (199 and 299)
  - » Therefore SPICE ephemeris objects 199 and 299 as well as 1 and 2 are found in a planet ephemeris file
- Because the masses of Phobos and Deimos are so small compared to the mass of Mars, the mass center for Mars (499) **was** treated as being located at the Mars barycenter (4)
  - » Starting in 2013 with the JPL planetary ephemeris named DE430 this is no longer the case; there is a very small offset of about 20 cm

- **For the solar system**

- Planet system barycenters (i.e. 1 through 9) and the sun (10) orbit the solar system barycenter (0)



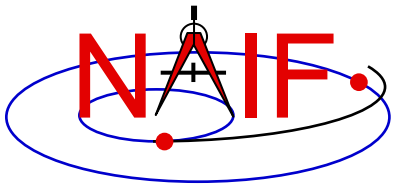
# Barycenter Offset Magnitude

Navigation and Ancillary Information Facility

| <u>Body Mass Center</u> | <u>System Barycenter</u> | <u>Barycenter offset from body mass center (km)*</u> | <u>Offset as % of body radius*</u> |
|-------------------------|--------------------------|------------------------------------------------------|------------------------------------|
| Sun (10)                | SSB (0)                  | 1,378,196                                            | 198%                               |
| Mercury (199)           | M. BC (1)                | 0                                                    | 0                                  |
| Venus (299)             | V. BC (2)                | 0                                                    | 0                                  |
| Earth (399)             | E. BC (3)                | 4942                                                 | 77%                                |
| Mars (499)              | M. BC (4)                | 0.0002                                               | ~ 0                                |
| Jupiter (599)           | J. BC (5)                | 220                                                  | 0.3%                               |
| Saturn (699)            | S. BC (6)                | 312                                                  | 0.5%                               |
| Uranus (799)            | U. BC (7)                | 43                                                   | 0.17%                              |
| Neptune (899)           | N. BC (8)                | 74                                                   | 0.3%                               |
| Pluto (999)**           | P. BC (9)                | 2080                                                 | 172%                               |

\* Estimated maximum values over the time range 2000-2050.

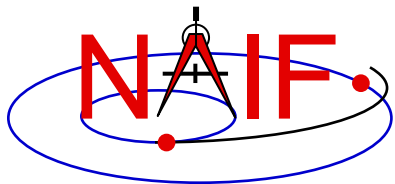
\*\* For ephemeris purposes, Pluto is still treated as a planet.



# SPK File Contents

Navigation and Ancillary Information Facility

- **A single SPK file can hold data for one ephemeris object, or for many ephemeris objects**
- **The ephemeris objects in a given SPK file need not all be of the same type**
  - One might find data for a spacecraft, some planets, and some satellites all in one file, split across multiple segments
- **This is illustrated in the next three charts**



# Examples of Generic SPK File Contents

## Navigation and Ancillary Information Facility

### Planet Ephemeris

### Asteroid Ephemeris

### Merged Planet<sup>4</sup> and Satellite Ephemeris

***0*** ***Solar System BC***  
**1** **Merc. BC**  
**199<sup>1</sup>** **Mercury**  
**2** **Venus BC**  
**299<sup>1</sup>** **Venus**  
**3** **Earth BC**  
**301<sup>2</sup>** **Moon**  
**399<sup>2</sup>** **Earth**  
**4** **Mars BC**  
**5** **Jupiter BC**  
**6** **Saturn BC**  
**7** **Uranus BC**  
**8** **Neptune BC**  
**9** **Pluto BC\***  
**10<sup>3</sup>** **Sun**

\* For ephemeris purposes, Pluto is still treated as a planet.

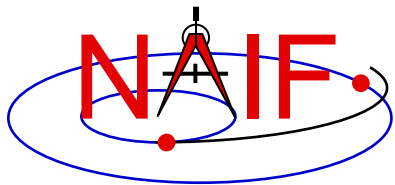
***10*** ***Sun***  
**2000001** **Ceres**

#### Notes:

- (1) Mercury and Venus planet locations are included in planet ephemerides since there are no satellite ephemerides for these planets.
- (2) The Moon and Earth locations are included in each planetary ephemeris because of historical ephemeris production techniques.
- (3) The Sun's location is included in each planetary ephemeris because of historical ephemeris production techniques.
- (4) For user convenience, NAIF often merges into a planet's satellite ephemeris files the locations of the earth, the earth barycenter and the sun.

***5*** ***Jupiter BC***  
**3<sup>4</sup>** **Earth BC**  
**10<sup>4</sup>** **Sun**  
**399<sup>4</sup>** **Earth**  
**501** **Io**  
**502** **Europa**  
**503** **Ganymede**  
**504** **Callisto**  
**505** **Amalthea**  
**514** **Thebe**  
**515** **Adrastea**  
**516** **Metis**  
**599** **Jupiter**

***The objects in blue italic font are the center of motion for the remaining objects in each file. There is no ephemeris data present for these centers of motion.***



# Examples of a Flight Project's SPK File Contents

Navigation and Ancillary Information Facility

This made-up example shows four collections of SPK files for the Cassini mission

## Cassini Orbiter

**6 = Saturn bc**

-82 = Cassini S/C

One object

## Huygens Probe

**6 = Saturn bc**

-150 = Huygens Probe

One object

## Planets

**0 = solar system bc**

3 = Earth barycenter  
6 = Saturn barycenter  
399 = Earth mass center  
301 = Moon

Multiple objects

## Satellites - 1

**6 = Saturn bc**

601 = Mimas  
602 = Enceladus  
603 = Tethys  
604 = Dione  
605 = Rhea  
606 = Titan  
607 = Hyperion  
608 = Iapetus  
609 = Phoebe  
699 = Saturn mass center

Multiple objects

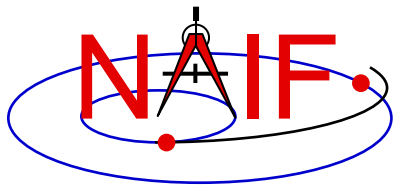
610 = Janus  
611 = Epimetheus  
:  
617 = Pandora  
699 = Saturn mass center

Multiple objects

## Satellites - 2

- The user's program must "load" as many of these SPK files as needed to satisfy her/his requirements.
- Sometimes a project NAV team combines (merges) several of these collections before releasing them, making the user's job easier.
- **Objects in blue font are the centers of motion for the remaining objects; these don't have ephemeris data included in the file.**

See the next page for a graphical representation of this collection of SPKs



# Possible\* SPK File Time Coverages for the Previous Example

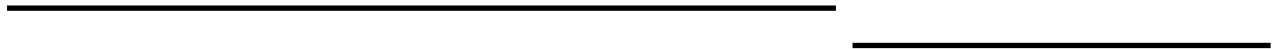
Navigation and Ancillary Information Facility

Each bar represents a separate file (SPK kernel)

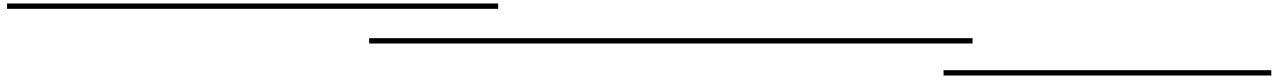
Planet:



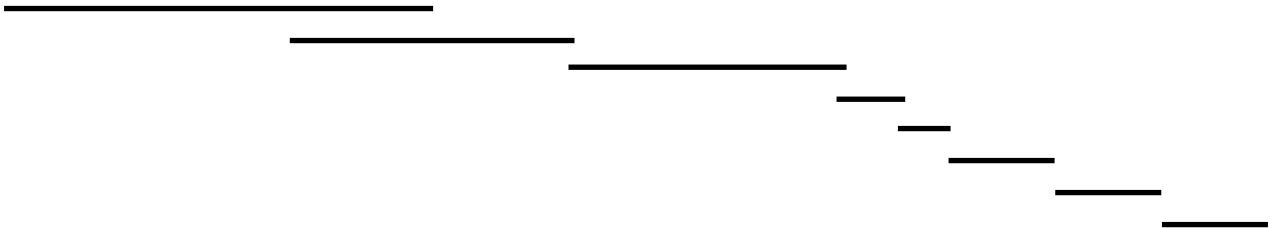
Satellite - 1:  
(Major satellites)



Satellite - 2:  
(Minor satellites)



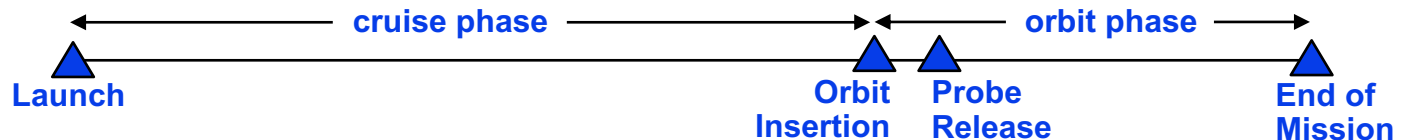
Orbiter :



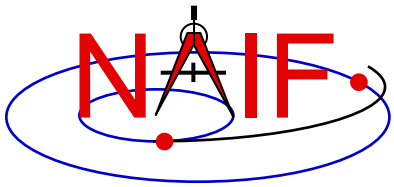
Huygens Probe :



Time line:



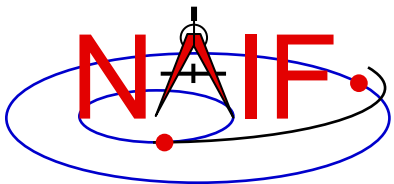
\* Note: This was not the real Cassini scenario—it is simply an illustration of some of the possibilities for ephemeris delivery on a planetary mission.



# SPKs for Objects Located on the Surface of a Natural Body

Navigation and Ancillary Information Facility

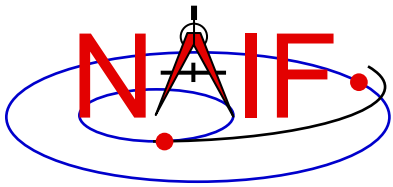
- **An SPK file may contain positions of tracking stations, observatories, rovers, etc.**
  - The object could be stationary or moving
  - Usually such SPKs contain ephemeris data given in the body-fixed reference frame
- **One reads this file the same as for any other SPK file**
  - Use the name or NAIF ID of the antenna, observatory or rover as the “target” or “observer” in an SPK reader argument list
  - Also requires use of a SPICE PCK file if you request vectors to be returned in an inertial frame such as J2000; the PCK is needed to rotate body-fixed vectors to the J2000 frame



---

**Navigation and Ancillary Information Facility**

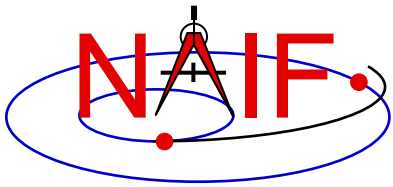
## **Using SPK Files**



# Retrieving Position or State Vectors

Navigation and Ancillary Information Facility

- **To retrieve position or state vectors of ephemeris objects one often needs two kinds of SPICE kernels**
  - Ephemeris kernel(s) (SPK)
  - Leapseconds kernel (LSK)
    - » The LSK is used to convert between Coordinated Universal Time (UTC) and Barycentric Dynamical Time (TDB, also called Ephemeris Time, ET)
      - This conversion is done outside of the SPK subsystem
- **Retrieving ephemeris data from an SPK file is usually called “reading” the file**
  - This term is not very accurate since the SPK “reader” software also performs interpolation, and may chain together data from multiple sources, do frame transformations, and perform aberration corrections
- **State and position vectors retrieved from an SPK file by the SPK “reader” routines are Cartesian (rectangular) vectors of the form:**
  - $X, Y, Z, dX, dY, dZ$  for a state vector (km, km/sec)
  - $X, Y, Z$  for a position vector (km)



# Retrieving a State Vector

Navigation and Ancillary Information Facility

Initialization...typically done once per program execution

Fortran syntax  
used here

Tell your program which SPICE files to use (“loading” files)

```
CALL FURNISH ('spk_file_name')
```

```
CALL FURNISH ('leapseconds_file_name')
```

It's better to replace these two calls with a single call to a “furnsh kernel” containing the names of all kernel files to load.

Loop... do as many times as you need to

Convert UTC time to ephemeris time (TDB), if needed

```
CALL STR2ET ( 'utc_string', tdb)
```

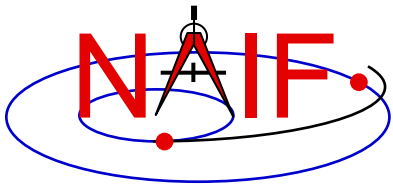
Retrieve state vector from the SPK file at your requested time

```
CALL SPKEZR (target, tdb, 'frame', 'correction', observer, state, light time)
```

inputs

outputs

Now use the returned state vector in other SPICE routines to compute observation geometry of interest.



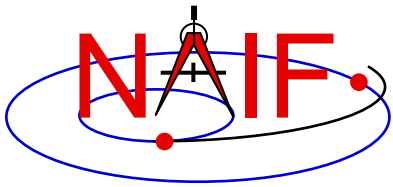
# Arguments of SPKEZR - 1

Navigation and Ancillary Information Facility

## INPUTS

- **TARGET and OBSERVER:** Character names\* or NAIF IDs for the end point and origin of the state vector (Cartesian position and velocity vectors) to be returned.
  - The position component of the requested state vector points from the observer to the target.
- **TDB:** The time at the observer's location at which the state vector is to be computed. The time system used is Ephemeris Time (ET), now generally called Barycentric Dynamical Time (TDB).
- **FRAME:** The SPICE name for the reference frame in which the output state vector is to be given. SPK software will automatically convert ephemeris data to the frame you specified, if needed. SPICE must know the named frame, either built-in or specified using a Frames Kernel.

\* Character names work for the target and observer inputs only if built into SPICE or if registered using the SPICE ID-body name mapping facility. Otherwise use the SPICE numeric ID enclosed in quotes.



# Arguments of SPKEZR - 2

Navigation and Ancillary Information Facility

- **CORRECTION:** Specification of what kind of aberration correction(s), if any, to apply in computing the output state vector.
  - Use LT+S to obtain the apparent state of the target as seen by the observer. LT+S invokes light time and stellar aberration corrections. (CN+S is better in some cases.)
  - Use NONE to obtain the uncorrected (aka “geometric”) state, as given by the source SPK file or files.

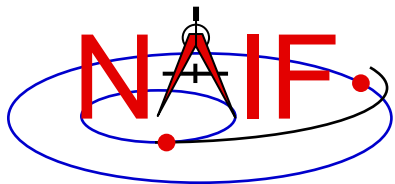
See the header for subroutine SPKEZR, the document SPK Required Reading, or the “Fundamental Concepts” tutorial for details. See the backup charts for examples of aberration correction magnitudes.

## OUTPUTS

- **STATE:** This is the Cartesian state vector you requested. It contains 6 components: three for position (x,y,z) and three for velocity (dx, dy, dz) of the target with respect to the observer. The position component of the state vector points from the *observer* to the *target*.
- **LIGHT TIME:** The one-way light time between the (optionally aberration-corrected) position of target and the geometric position of the observer at the specified epoch. (Generally not needed.)

LT + S = light time plus stellar aberration

CN + S = converged Newtonian light time plus stellar aberration



# A Simple Example of Retrieving a State Vector

Navigation and Ancillary Information Facility

*Fortran syntax  
used here*

Initialization - typically do this just **once** per program execution

```
CALL FURNISH ( 'NAIF0012.TLS' )  
CALL FURNISH ( 'CASSINI_MERGED.BSP' )
```

It's better to replace these two calls with a single call loading a "furnsh kernel" containing the names of all kernel files to load.

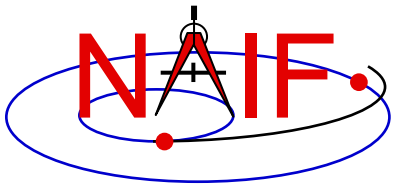
Repeat in a loop if/as needed to solve your particular problem

```
CALL STR2ET ( '2004 NOV 21 02:40:21.3', TDB )  
CALL SPKEZR ( 'TITAN', TDB, 'J2000', 'LT+S', 'CASSINI',  
              STATE, LT )
```

(Insert additional code here to make derived computations such as spacecraft sub-latitude and longitude, lighting angles, etc. Use more SPICE subroutines to help.)

In this example we get the state (STATE) of Titan as seen from the Cassini spacecraft at the UTC epoch 2004 NOV 21 02:40:21.3. The state vector is returned in the J2000 inertial reference frame, which in SPICE is the same as the ICRF frame. The state vector has been corrected for both light time and stellar aberration (LT+S). The one-way light time (LT) is also returned, just in case it could be useful.

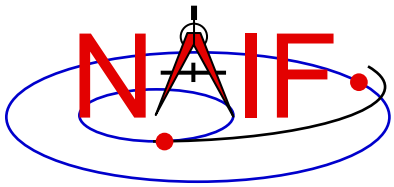
A SPICE leapseconds file (NAIF0012.TLS) is used, as is a SPICE ephemeris file (CASSINI\_MERGED.BSP) containing ephemeris data for Cassini (-82), Saturn barycenter (6), Saturn mass center (699), Saturn's satellites (6xx) and the sun (10).



# Retrieving a Position Vector

Navigation and Ancillary Information Facility

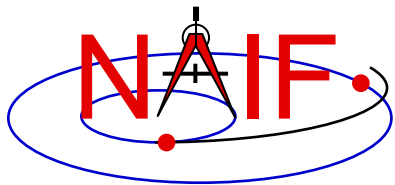
- **SPKPOS is the position-only analog of SPKEZR**
  - The arguments of SPKPOS are identical to those of SPKEZR, except that SPKPOS returns a 3-component position vector instead of a 6-component state vector
  - SPKPOS executes more quickly than SPKEZR when stellar aberration corrections are used
  - SPKPOS can be used when reference frame transformations of velocity are not possible due to absence of C-kernel angular velocity data



# A Slightly More Complex Example Kernel Data Needed

Navigation and Ancillary Information Facility

- To get state vectors referenced to a non-inertial reference frame, or when the data within the SPK file are provided in a non-inertial frame, typically more kernels will be needed.
  - To get the state of an object relative to a body in the body's **IAU body-fixed reference frame** you'll need:
    - » PCK file containing orientation data for the body
    - » SPK(s) for the object and body
    - » LSK
  - To get the state of an object in a **spacecraft-fixed reference frame** you'll need:
    - » FK, CK and SCLK for the spacecraft
    - » SPK(s) for the spacecraft and object
    - » LSK



# A Slightly More Complex Example Retrieving a State Vector

Navigation and Ancillary Information Facility

Obtain the state of Titan relative to Cassini in the Titan body-fixed reference frame

Initialization...typically **once** per program execution

Tell your program which SPICE files to use (“loading” files)

```
CALL FURNISH ('CASSINI_MERGED.BSP')  
CALL FURNISH ('NAIF0012.TLS')  
CALL FURNISH ('NAIF0011.TPC')
```

It's better to replace these three calls with a single call loading a “furnsh kernel” containing the names of all kernel files to load.

Fortran syntax  
used here

Loop... do as many times as you need

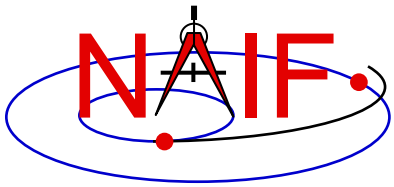
Convert UTC time to ephemeris time (TDB), if needed

```
CALL STR2ET ('2004 NOV 21 02:40:21.3', TDB)
```

Get state vector from SPK file at requested time, in satellite's IAU body-fixed frame

```
CALL SPKEZR ('TITAN', TDB, 'IAU_TITAN', 'LT+S', 'CASSINI',  
STATE, LT)
```

Insert additional code here to make derived computations such as spacecraft sub-latitude and longitude, lighting angles, etc. Use more SPICE subroutines to help.

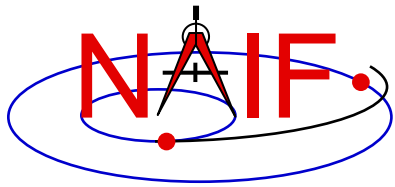


# Constant-velocity objects: Additional SPK State Computation APIs

---

Navigation and Ancillary Information Facility

- **The SPK subsystem contains routines for computing the state of an ephemeris object with respect to a fixed point or one moving with constant, non-zero velocity, in a specified reference frame.**
  - The point may act as either the target or the observer.
  - The center of motion of the point must be an ephemeris object.
- **The SPK routines providing this capability are:**
  - SPKCPT (SPK, constant position target)
  - SPKCPO (SPK, constant position observer)
  - SPKCVT (SPK, constant velocity target)
  - SPKCVO (SPK, constant velocity observer)
- **These routines may provide a convenient alternative to creating SPK files for surface points:**
  - when there's a need to compute the locations on the fly
  - when the number of surface points is large

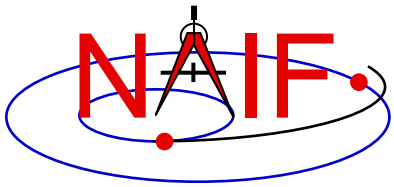


# Manipulating and Using SPK Files

Navigation and Ancillary Information Facility

- **You can subset an SPK, or merge two or more SPKs**
  - The subset or merge may be keyed off of objects, or time, or both
  - Merging only portions of SPKs is possible
- **You can read data from just one, or many\* SPK files in your application program**
  - Don't forget the precedence rule: data in a later loaded file take precedence over data from an earlier loaded file
- **You can convert an SPK that is in non-native binary format to native binary format if you need to add data or comments**

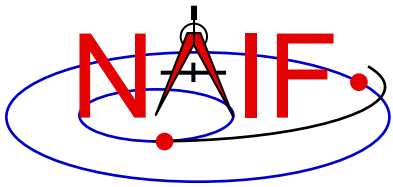
\* The allowed number of simultaneously loaded DAF- and DAS-based files is set to 5000 in N66 Toolkits. "DAF" is the acronym for Double Precision Array File. SPKs are based on the DAF architecture.



# Understanding an SPK File

Navigation and Ancillary Information Facility

- **The SPK producer should have provided descriptive meta-data inside an SPK file, in the “comment area”**
  - The comments should say when, why, how and for what purpose the file was made
  - Additional useful information could also be provided by the producer
    - » **Example: when and why any data gaps are present**
- **These comments may be extracted using an API (subroutine) or viewed using a SPICE utility program.**
  - API: DAFEC
  - Utility program: `commnt -r <spk_file_name>`



# SPK Utility Programs

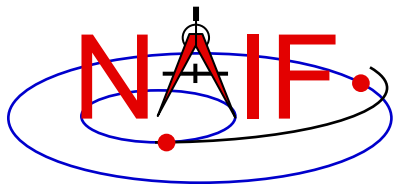
Navigation and Ancillary Information Facility

- **The following SPK utility programs are included in the Toolkit:**

|                 |                                                                  |
|-----------------|------------------------------------------------------------------|
| <b>BRIEF</b>    | summarizes coverage for one or more SPK files                    |
| <b>SPACIT</b>   | generates segment-by-segment summary of an SPK file              |
| <b>COMMNT</b>   | reads, appends, or deletes comments in an SPK file               |
| <b>MKSPK</b>    | converts ephemeris data provided in a text file into an SPK file |
| <b>SPKDIFF</b>  | compares two SPK files                                           |
| <b>SPKMERGE</b> | subsets an spk, or merges one or more SPK files or pieces        |

- **These additional SPK utility programs are provided on the NAIF Web site (<http://naif.jpl.nasa.gov/naif/utilities.html>)**

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| <b>SPY</b>      | validates, inspects, and analyses SPK files                    |
| <b>PINPOINT</b> | creates an SPK file for fixed locations (ground stations, etc) |
| <b>BSPIDMOD</b> | alters body IDs in an SPK file                                 |
| <b>DAFMOD</b>   | alters body or frame IDs in an SPK file                        |
| <b>DAFCAT</b>   | concatenates together SPK files                                |
| <b>BFF</b>      | displays binary file format of an SPK file                     |
| <b>BINGO</b>    | converts SPK files between big- and little-endian formats      |



# Summarizing an SPK File - 1

## Navigation and Ancillary Information Facility

- A summary of the contents and time coverage of an SPK file can be made using the SPICE Toolkit utility *“brief”*
  - See the brief User’s Guide for details

```
% brief 070413BP_SCPSE_07097_07121.bsp
```

Summary for: 070413BP\_SCPSE\_07097\_07121.bsp

|                        |                      |                |
|------------------------|----------------------|----------------|
| Bodies: CASSINI (-82)  | PLUTO BARYCENTER (9) | TETHYS (603)   |
| MERCURY BARYCENTER (1) | SUN (10)             | DIONE (604)    |
| VENUS BARYCENTER (2)   | MERCURY (199)        | RHEA (605)     |
| EARTH BARYCENTER (3)   | VENUS (299)          | TITAN (606)    |
| MARS BARYCENTER (4)    | MOON (301)           | HYPERION (607) |
| JUPITER BARYCENTER (5) | EARTH (399)          | IAPETUS (608)  |
| SATURN BARYCENTER (6)  | MARS (499)           | PHOEBE (609)   |
| URANUS BARYCENTER (7)  | MIMAS (601)          | SATURN (699)   |
| NEPTUNE BARYCENTER (8) | ENCELADUS (602)      |                |

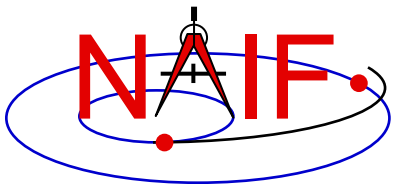
Start of Interval (ET)

-----  
2007 APR 07 16:22:23.000

End of Interval (ET)

-----  
2007 MAY 01 09:34:03.000

Note, the default  
time system is  
ET, not UTC!



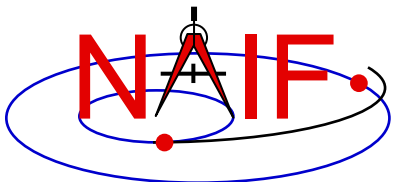
# Summarizing an SPK File - 2

## Navigation and Ancillary Information Facility

- Use of the “-c” option when using the *brief* utility will show you the center of motion for each object
  - This is often useful in diagnosing an SPK chaining problem
    - » See the “Problems” section at the end of this tutorial for more information

```
% brief -c 070413BP_SCPSE_07097_07121.bsp
```

```
Bodies: CASSINI (-82) w.r.t. SATURN BARYCENTER (6)
MERCURY BARYCENTER (1) w.r.t. SOLAR SYSTEM BARYCENTER (0)
VENUS BARYCENTER (2) w.r.t. SOLAR SYSTEM BARYCENTER (0)
EARTH BARYCENTER (3) w.r.t. SOLAR SYSTEM BARYCENTER (0)
MARS BARYCENTER (4) w.r.t. SOLAR SYSTEM BARYCENTER (0)
JUPITER BARYCENTER (5) w.r.t. SOLAR SYSTEM BARYCENTER (0)
SATURN BARYCENTER (6) w.r.t. SOLAR SYSTEM BARYCENTER (0)
URANUS BARYCENTER (7) w.r.t. SOLAR SYSTEM BARYCENTER (0)
NEPTUNE BARYCENTER (8) w.r.t. SOLAR SYSTEM BARYCENTER (0)
PLUTO BARYCENTER (9) w.r.t. SOLAR SYSTEM BARYCENTER (0)
SUN (10) w.r.t. SOLAR SYSTEM BARYCENTER (0)
MERCURY (199) w.r.t. MERCURY BARYCENTER (1)
VENUS (299) w.r.t. VENUS BARYCENTER (2)
MOON (301) w.r.t. EARTH BARYCENTER (3)
EARTH (399) w.r.t. EARTH BARYCENTER (3)
MARS (499) w.r.t. MARS BARYCENTER (4)
MIMAS (601) w.r.t. SATURN BARYCENTER (6)
ENCELADUS (602) w.r.t. SATURN BARYCENTER (6)
TETHYS (603) w.r.t. SATURN BARYCENTER (6)
DIONE (604) w.r.t. SATURN BARYCENTER (6)
RHEA (605) w.r.t. SATURN BARYCENTER (6)
TITAN (606) w.r.t. SATURN BARYCENTER (6)
HYPERION (607) w.r.t. SATURN BARYCENTER (6)
IAPETUS (608) w.r.t. SATURN BARYCENTER (6)
PHOEBE (609) w.r.t. SATURN BARYCENTER (6)
SATURN (699) w.r.t. SATURN BARYCENTER (6)
Start of Interval (ET)                                End of Interval (ET)
-----
2007 APR 07 16:22:23.000                                2007 MAY 01 09:34:03.000
```



# Summarizing an SPK File - 3

## Navigation and Ancillary Information Facility

- A detailed summary of an SPK can be made using the Toolkit utility named “*SPACIT*”
- See the *SPACIT* User’s Guide for details

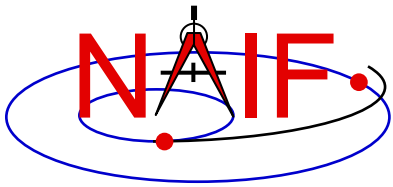
```
Summary for SPK file: sat240.bsp
Leapseconds File      : /kernels/gen/lsk/leapseconds.ker
Summary Type          : Entire File
```

```
-----
Segment ID           : SAT240
Target Body          : Body 601, MIMAS
Center Body          : Body 6, SATURN BARYCENTER
Reference frame:      Frame 1, J2000
SPK Data Type         : Type 3
  Description         : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
UTC Start Time        : 1969 DEC 31 00:00:00.000
UTC Stop Time         : 2019 DEC 02 00:00:00.000
ET Start Time         : 1969 DEC 31 00:00:41.183
ET Stop time          : 2019 DEC 02 00:01:05.183
-----
```

```
-----
Segment ID           : SAT240
Target Body          : Body 602, ENCELADUS
Center Body          : Body 6, SATURN BARYCENTER
Reference frame:      Frame 1, J2000
SPK Data Type         : Type 3
  Description         : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
UTC Start Time        : 1969 DEC 31 00:00:00.000
UTC Stop Time         : 2019 DEC 02 00:00:00.000
ET Start Time         : 1969 DEC 31 00:00:41.183
ET Stop time          : 2019 DEC 02 00:01:05.183
-----
```

:

(This is a partial output; not all data could be displayed on this chart)



# Summarizing an SPK File - 4

Navigation and Ancillary Information Facility

## Summarizing an SPK at the API Level

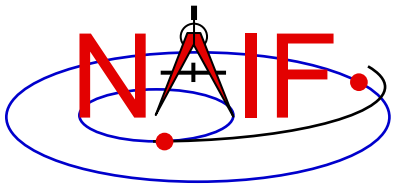
- **Call SPKOBJ to find the set of objects for which a specified SPK provides data.**
  - **INPUT:** an SPK file name and initialized SPICE integer “Set” data structure. The set may optionally contain ID codes obtained from previous calls.
  - **OUTPUT:** the input set, to which have been added (via set union) the ID codes of objects for which the specified SPK provides coverage.

```
CALL SPKOBJ ( SPK, IDSET )
```

- **Call SPKCOV to find the window of times for which a specified SPK file provides coverage for a specified body:**
  - **INPUT:** an SPK file name, body ID code and initialized SPICE double precision “Window” data structure. The window may optionally contain coverage data from previous calls.
  - **OUTPUT:** the input window, to which have been added (via window union) the sequence of start and stop times of segment coverage intervals of the specified SPK, expressed as seconds past J2000 TDB.

```
CALL SPKCOV ( SPK, IDCODE, COVER )
```

- **See the headers of these routines for example programs.**
- **Also see the CELLS, SETS and WINDOWS Required Reading for background information on these SPICE data types.**

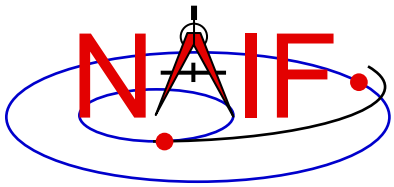


# Additional Information on SPK

---

Navigation and Ancillary Information Facility

- **For more information about SPK, look at the following:**
  - The remainder of this tutorial (the BACKUP section)
  - Most Useful Routines document
  - SPK Required Reading document
  - Headers of the subroutines mentioned
  - Using Frames tutorial
  - BRIEF and SPKDIFF User's Guides
- **Related documents:**
  - NAIF\_IDS Required Reading
  - Frames Required Reading
  - Time Required Reading
  - Kernel Required Reading

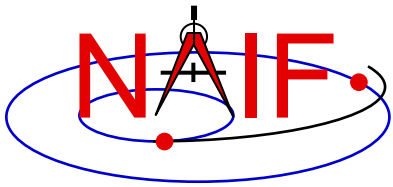


# Backup

---

Navigation and Ancillary Information Facility

- **Problems Using SPK Files**
- **Don't Mix Planet Ephemerides**
- **Effect of Aberration Corrections**
- **Examples of Retrieving State Vectors**
- **SPK File Structure**



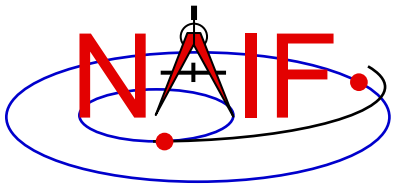
# Problems Using SPK Files - 1

Navigation and Ancillary Information Facility

- The file, or files, you loaded do not contain data for both your target and observer bodies
  - You may have loaded the wrong file, or assumed the file contains data that it doesn't
  - You may not have loaded all the files needed
- The file, or files, you loaded do not cover the time at which you requested a state vector
  - This could occur if you've been given a file coverage summary in calendar ET form and you mistook this for UTC  
(  $ET = UTC + DELTAET$ , where  $DELTAET$  is about 69 seconds as of 1/20)
  - This could occur if you are requesting a light-time corrected state and the SPK files being used do not have data at the time that is one-way light-time away\* from your ET epoch of interest
    - » \* Earlier, for the receive case; later, for the transmission case
- In the above situations you'll get an error message like the following:

```
SPICE (SPKINSUFFDATA) - -
```

```
Insufficient ephemeris data has been loaded to  
compute the state of xxx relative to yyy.
```



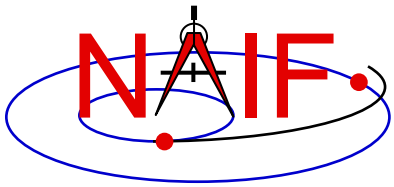
# Problems Using SPK Files - 2

Navigation and Ancillary Information Facility

- **You have requested aberration-corrected states but the SPK file or files you loaded do not contain sufficient data to relate both your target and observer bodies back to the solar system barycenter, which is required for this calculation.**
  - You may not have loaded all the files needed
  - You may have assumed the file contains data that it doesn't
- **In the above situations you'll get an error message like the following:**

```
SPICE (SPKINSUFFDATA) - -
```

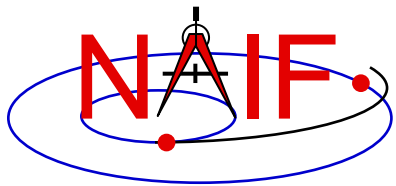
```
Insufficient ephemeris data has been loaded to  
compute the state of xxx relative to yyy.
```



# Problems Using SPK Files – 3a

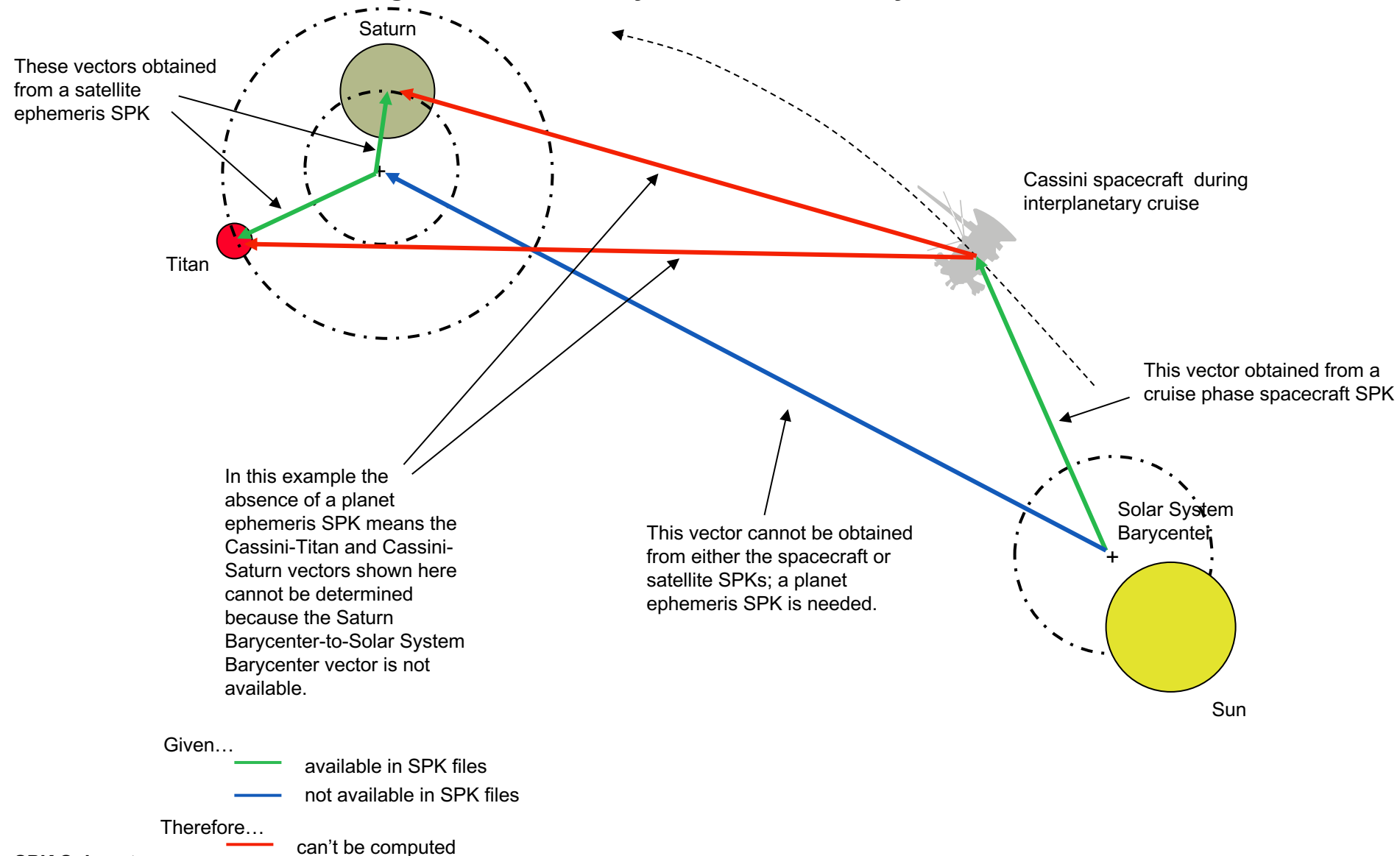
Navigation and Ancillary Information Facility

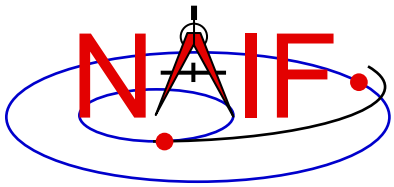
- **An infrequent problem occurs when your SPK file(s) contain data for both target and observer, and cover the period of interest, but ephemeris data for an intermediate body needed to chain the target and observer together is missing.**
  - **Example: You load a spacecraft SPK containing ephemeris for Cassini (-82) relative to the solar system barycenter (0), and you load a satellite SPK containing the ephemeris for Titan (606) and Saturn (699) relative to the Saturn barycenter (6). But you forgot to load a planet SPK file that contains data for the Saturn barycenter (6) relative to the solar system barycenter (0). The SPK software cannot “connect” Cassini to Titan or to Saturn. (See the drawing on the next page.)**
  - **In this case, knowing what is the “Center Body” of movement for each target body is important; this is shown in SPACIT, and also in BRIEF summaries when the -c command line option is used.**
  - **This problem is illustrated on the next page.**



# Problems Using SPK Files - 3b (drawing)

## Navigation and Ancillary Information Facility



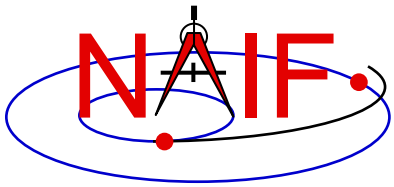


# Problems Using SPK Files - 4

---

Navigation and Ancillary Information Facility

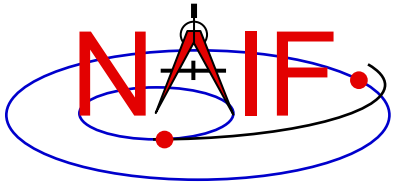
- **You see an error message to the effect that pole RA (right ascension) data cannot be found**
  - **You are requesting results in a body-fixed frame, but you have not loaded a SPICE PCK file that defines this frame.**



# Problems Using SPK Files - 5

Navigation and Ancillary Information Facility

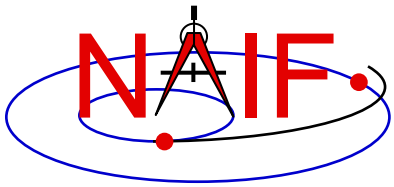
- **Segment Masking:** You've loaded sufficient data to chain together the target and observer, but the SPK subsystem can't make the connection.
  - This can happen when a high-priority segment that can't be connected to both target and observer "masks" a lower-priority segment that can be connected.
  - Example: you want the state of earth as seen from the Galileo orbiter at a specified ephemeris time ET1.
    - » You have loaded SPK files providing:
      - the state of the Galileo orbiter relative to the asteroid Gaspra
      - the state of the orbiter relative to the sun
      - the state of the earth relative to the earth-moon barycenter
      - the states of the sun and earth-moon barycenter relative to the solar system barycenter
    - » If an SPK segment for the orbiter relative to Gaspra covering ET1 has higher priority than the segment for the orbiter relative to the sun covering ET1, no connection between the orbiter and the earth will be made.
    - » Solution:
      - Load an SPK file providing the ephemeris of Gaspra relative to the sun or the solar system barycenter (for a time interval containing ET1)



# Problems Using SPK Files - 6

Navigation and Ancillary Information Facility

- **Other missing data... not obvious.**
  - You may need CK (and SCLK), FK or PCK data to construct your requested output frame.
- **Mistaking ET for UTC, or vice-versa: very common**
- **Using light time corrections requires target ephemeris data at the light time-corrected epoch.**
  - If you're working near the beginning of an SPK file, the light time-corrected epoch may occur earlier than available data.



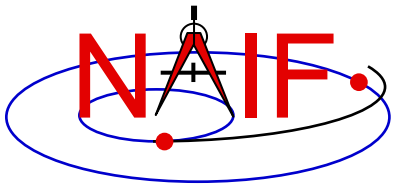
# Problems Using SPK Files - 7

Navigation and Ancillary Information Facility

- You've assumed that:

$$\text{state (observer, target)} = \overset{\text{negative sign}}{-} \text{state (target, observer)}$$

- This is NOT true unless you have requested geometric states in both cases (i.e. no light time or stellar aberration corrections are applied)

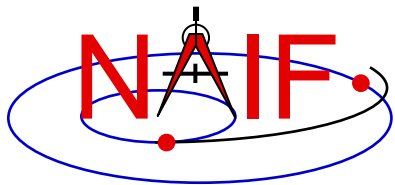


# Problems Using SPK Files - 8

---

Navigation and Ancillary Information Facility

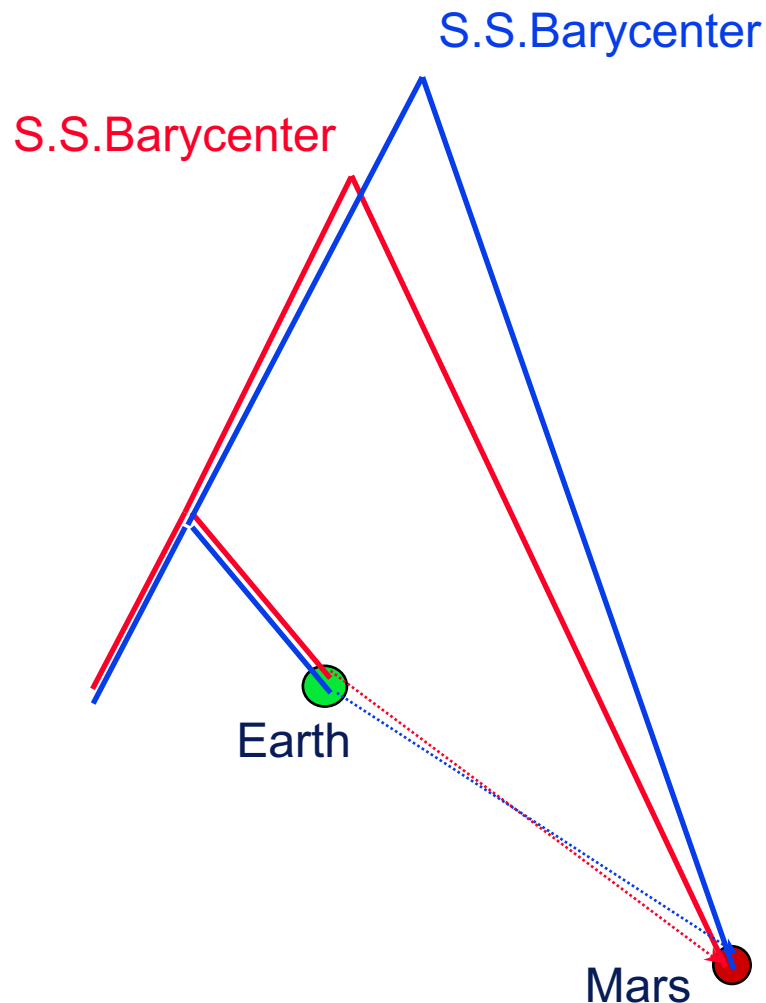
- **SPK reading efficiency can be impacted by any of several circumstances.**
  - This won't result in erroneous results... just slow performance.
  - This sort of problem does not occur very often.
  - For details, read some of the backup charts in the tutorial named Making an SPK—those titled "SPK Reading: Efficiency Issues."

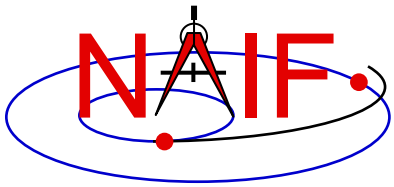


# Don't Mix Planet Ephemerides-1

Navigation and Ancillary Information Facility

- With each new version of the JPL planetary ephemeris, the solar system barycenter moves with respect to the planets
- Changes in relative planet positions are much smaller than changes in the planet locations relative to the solar system barycenter

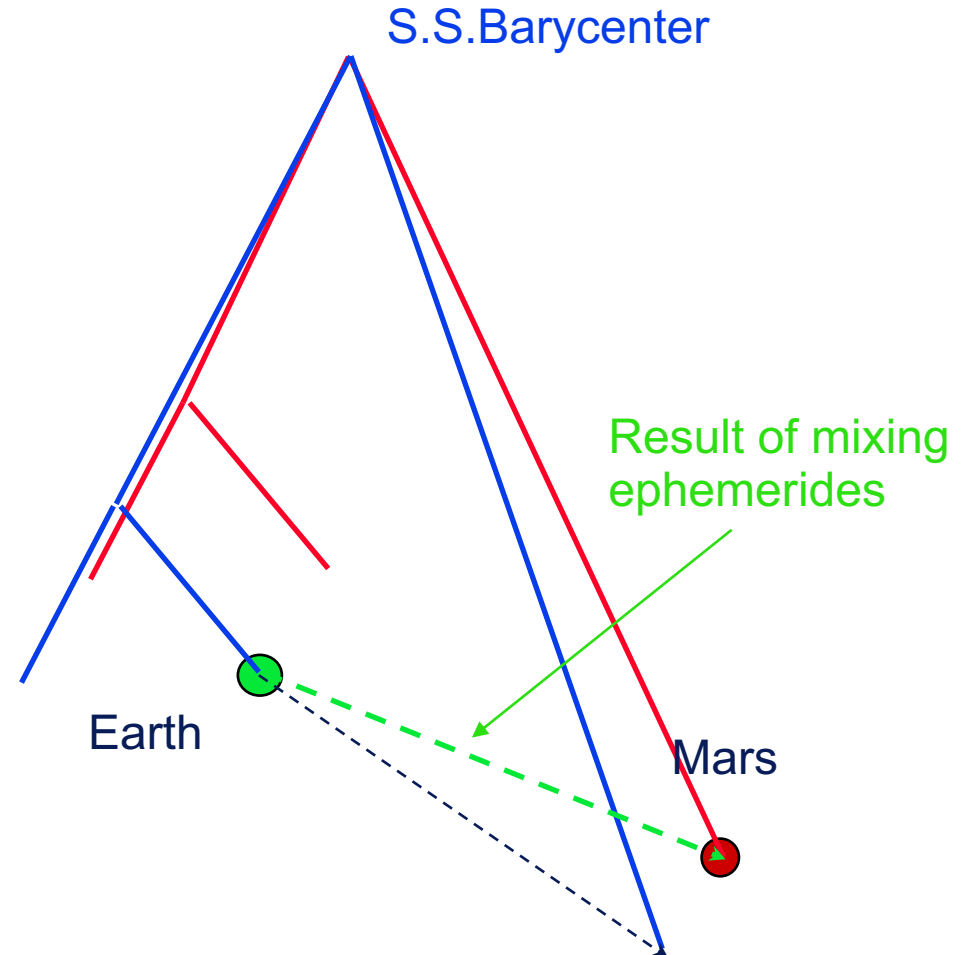


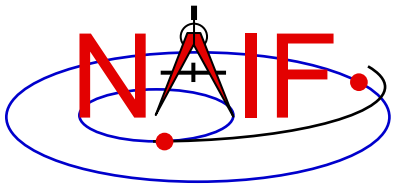


# Don't Mix Planet Ephemerides-2

Navigation and Ancillary Information Facility

- **SPICE allows you to “load” different planetary ephemerides (or portions of them)**
  - » You can potentially subtract the solar system barycenter-relative positions from different ephemerides to get relative states
- **Don't mix planetary ephemerides**
- **For JPL flight projects, a consistent set of ephemerides is provided to the mission team**

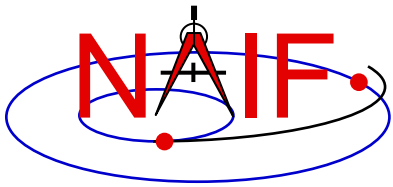




# Effect of Aberration Corrections - 1

Navigation and Ancillary Information Facility

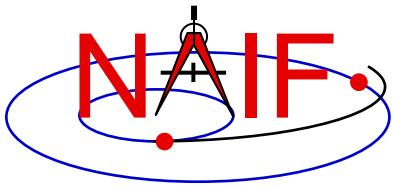
- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1 to 2005 Jan1**
  - **Mars as seen from MEX:**
    - » **LT+S vs NONE: .0002 to .0008 degrees**
    - » **LT vs NONE: .0006 to .0047 degrees**
  - **Earth as seen from MEX:**
    - » **LT+S vs NONE: .0035 to .0106 degrees**
    - » **LT vs NONE: .0000 to .0057 degrees**
  - **MEX as seen from Earth:**
    - » **LT+S vs NONE: .0035 to .0104 degrees**
    - » **LT vs NONE: .0033 to .0048 degrees**
  - **Sun as seen from Mars:**
    - » **LT+S vs NONE: .0042 to .0047 degrees**
    - » **LT vs NONE: .0000 to .0000 degrees**



# Effect of Aberration Corrections - 2

Navigation and Ancillary Information Facility

- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1 to 2008 Jan1**
  - **Saturn as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0058 degrees**
    - » **LT vs NONE: .0001 to .0019 degrees**
  - **Titan as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0057 degrees**
    - » **LT vs NONE: .0000 to .0030 degrees**
  - **Earth as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0107 degrees**
    - » **LT vs NONE: .0000 to .0058 degrees**
  - **CASSINI as seen from Earth:**
    - » **LT+S vs NONE: .0000 to .0107 degrees**
    - » **LT vs NONE: .0000 to .0059 degrees**
  - **Sun as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0059 degrees**
    - » **LT vs NONE: .0000 to .0000 degrees**

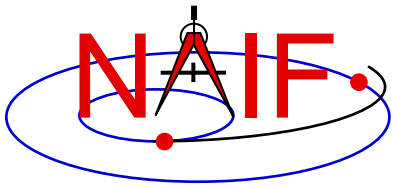


# Examples of Retrieving State Vectors – 1

## Navigation and Ancillary Information Facility

- **Example: find the geometric state of the MGS orbiter relative to Mars at the observation epoch ET, expressed in the J2000 reference frame.**
  - **CALL SPKEZR ( 'MGS', ET, 'J2000', 'NONE', 'MARS', STATE, LT )**
  - The SPK subsystem locates an SPK segment containing the ephemeris of the orbiter relative to Mars covering epoch ET, interpolates the ephemeris data at ET, and returns the interpolated state vector.
- **Example: find the geometric state of Titan relative to the earth at epoch ET, expressed in the J2000 reference frame.**
  - **CALL SPKEZR ( 'TITAN', ET, 'J2000', 'NONE', 'EARTH', STATE, LT )**
  - The SPK subsystem looks up and interpolates ephemeris data to yield:
    - » The state of the earth relative to the earth-moon barycenter (A)
    - » The state of the earth-moon barycenter relative to the solar system barycenter (B)
    - » The state of Titan relative to the Saturn system barycenter at ET (C)
    - » The state of the Saturn system barycenter relative to the solar system barycenter at ET (D)
  - SPKEZR then returns the state vector
    - »  $C + D - (A + B)$

Fortran syntax  
used here

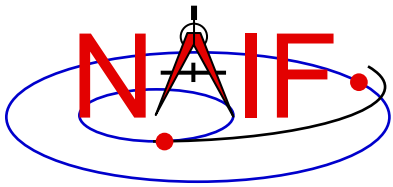


## Examples of Retrieving State Vectors – 2

Navigation and Ancillary Information Facility

- **Example: find the apparent state of the Cassini orbiter relative to the DSN station DSS-14, expressed in the topocentric reference frame centered at DSS-14, at a specified observation epoch ET.**
  - **CALL SPKEZR ( 'CASSINI', ET, 'DSS-14\_TOPO',  
'LT+S', 'DSS-14', STATE, LT )**
  - **The SPK subsystem looks up and interpolates ephemeris data to yield:**
    - » **The state of DSS-14 relative to the earth in the ITRF93 terrestrial reference frame (A)**
    - » **The state at ET of the earth relative to the earth-moon barycenter in the J2000 reference frame (B)**
    - » **The state at ET of the earth-moon barycenter relative to the solar system barycenter in the J2000 frame (C)**
    - » **The state at the light time-corrected epoch ET-LT of the Cassini orbiter relative to the Saturn system barycenter (other centers are possible) in the J2000 frame (D)**

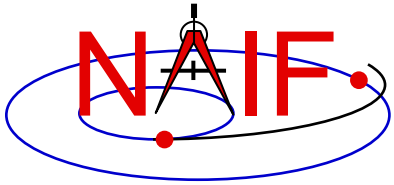
*continued on next page*



## Examples of Retrieving State Vectors – 3

### Navigation and Ancillary Information Facility

- » The state at ET-LT of the Saturn system barycenter relative to the solar system barycenter in the J2000 frame (E)
- The SPK subsystem also looks up transformation matrices to map states:
  - » From the J2000 frame to the ITRF93 terrestrial (earth body-fixed) frame at the observation epoch ET (T1)
  - » From the ITRF93 terrestrial frame to the DSS-14-centered topocentric frame (T2)
- SPKEZR then computes the J2000-relative, light-time corrected observer-target state vector
  - »  $E + D - ((T1)^{-1} * A + B + C)$
- SPKEZR corrects this vector for stellar aberration
  - » Call the result "V\_J2000\_apparent"
- and finally returns the requested state vector in the DSS-14 topocentric reference frame
  - »  $STATE = T2 * T1 * V\_J2000\_apparent$



# SPK File Structure

---

Navigation and Ancillary Information Facility

- **A description of the SPK file structure is shown near the beginning of the “Making an SPK” tutorial.**